
195 PTAS.
(IVA Incluido)
108
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
P.V.P. Canarias, Ceuta y Melilla 185 Ptas.
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IX - Fascículo 108

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-181-X (tomo 9)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 118602
Impreso en España-Printed in Spain-Febrero 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Recorrer el circuito

Iniciamos una serie dedicada a analizar los componentes de las placas de circuito impreso de diversos micros. En primer lugar, centraremos nuestra atención en el microprocesador

Ampliación
Esta fotomicrografía es una imagen muy ampliada de un chip Zilog Z80, utilizado actualmente, entre otros, en el Spectrum y la gama Amstrad CPC. El Z80 posee más de 25 registros en el chip, que aparecen como formatos rectangulares. Se pueden apreciar las conexiones del bus; alrededor del perímetro, 40 rellenos de conexión unen el chip con las patillas que permiten montarlo en una placa de circuito impreso

Mirando una fotomicrografía de un microprocesador típico percibimos cuadrículas geométricas, que son los registros internos del procesador, varias pistas rectangulares alrededor de la parte exterior, donde se realizan las conexiones externas, y zonas de sistemas de circuitos lógicos entremezclados.

También es posible discernir grupos de líneas que son los enlaces de comunicación interna entre los diversos componentes del chip. Es posible apreciar que la arquitectura de un procesador está determinada por las restricciones de un trazado bidimensional y por la cantidad física de componentes que se pueden incluir en un chip.

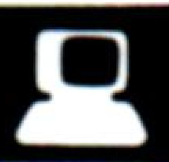
Lo que en realidad hacen las instrucciones del procesador es poner en funcionamiento una cadena de "microeventos" electrónicos, y éste es el punto en donde se traza la línea divisoria entre software y hardware. Vamos a ilustrar esto tomando el ejemplo de una instrucción sencilla: sumar un número al valor del acumulador y volver a colocar el resultado en el acumulador. El diagrama (abajo) muestra un ejemplo clásico de arquitectura de procesador de una forma simplificada. Todos los registros internos están unidos mediante un bus de datos interno y cada registro tiene conectadas líneas de control de modo que todas las operaciones de transferencia de datos entre registros y el bus de datos están sincronizadas. Estas líneas de control se unen en una uni-

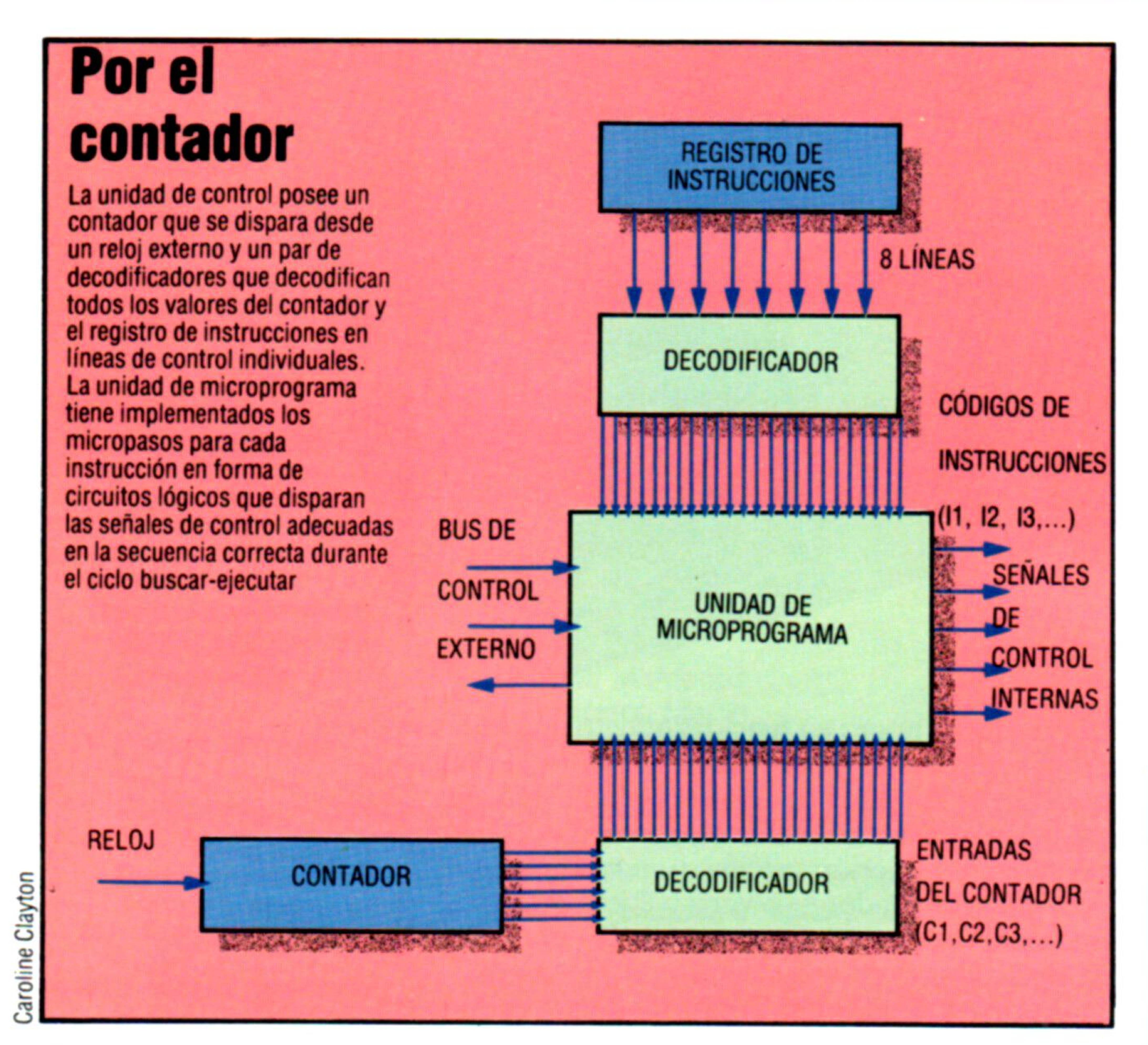
El procesador
El diagrama muestra la arquitectura típica de un procesador de ocho bits. Los componentes en el chip se comunican a través de un bus de datos interno y señales de control internas, pero se mantienen en contacto con la memoria y otros dispositivos mediante señales de control y buses de datos y direcciones externos

Arquitectura a pequeña escala

BUS DE DATOS INTERNO
BUS DE DATOS EXTERNO
REGISTRO DE DATOS DE MEMORIA (M)
EI EO
ACUMULADOR (A)
EI EO
REGISTRO TEMPORAL (T)
EO EI
REGISTRO DE ESTADO
EI EO
REGISTRO DE INSTRUCCIONES (I)
EI EO EOC
CONTADOR DEL PROGRAMA (PC)
EO +1 EI EOA
EO
ALU
BUS DE CONTROL EXTERNO
RELOJ
UNIDAD DE CONTROL
SEÑALES DE CONTROL INTERNAS
BUS DE DIRECCIONES EXTERNO

Paul Bryant





Por el contador

La unidad de control posee un contador que se dispara desde un reloj externo y un par de decodificadores que decodifican todos los valores del contador y el registro de instrucciones en líneas de control individuales. La unidad de microprograma tiene implementados los micropasos para cada instrucción en forma de circuitos lógicos que disparan las señales de control adecuadas en la secuencia correcta durante el ciclo buscar-ejecutar

Caroline Clayton

Circuito de micropasos
Para nuestro ejemplo de instrucción ADD, es necesario establecer, en el paso 2 y el paso 7 del ciclo buscar-ejecutar, una señal de control determinada, la habilitación de salida en el registro de datos de memoria. Este circuito lógico simple combina la señal 206 del decodificador de instrucciones con las dos señales de contador decodificadas adecuadas para disparar "EO on M"

I206
C7
C2
EO ON M

dad de control central, de la que luego hablaremos con mayor detalle.

La instrucción ADD (suma) inmediata que mencionábamos anteriormente cobra vida en forma de un par de bytes en un programa más grande almacenado en la memoria del ordenador. El primer byte contiene la instrucción y el segundo, el número a sumar. Para obedecer esta instrucción, el procesador primero debe traer de la memoria el byte de instrucción y después ejecutarlo: el ciclo buscar-ejecutar. El procesador sabe dónde obtener la instrucción, dado que su dirección estará retenida en el contador del programa (PC: *program counter*). Al describir el ciclo buscar-ejecutar, nos interesa particularmente la parte relativa a cómo gestiona la unidad de control las diversas transferencias entre registros. Esto se realiza enviándole señales *enable* (de habilitación) al registro correcto en el momento adecuado. El sistema que hemos utilizado para diferenciar estas señales es el siguiente:

EI: habilitar entrada a registro desde bus de datos interno
EO: habilitar salida desde un registro hacia el bus de datos
EOC: habilitar salida a la unidad de control
EOA: habilitar salida al bus de direcciones

El ciclo "buscar" comprende estos pasos:

Paso	Señales control	Acción
1	EOA on PC	Copiar el cont. del PC en el bus de direcciones
2	+1 on PC	Incrementar el PC
3	EO on M EI on I	Copiar el contenido del registro de datos de memoria en el registro de instrucciones
4	EOC on I	Copiar el contenido del registro de instrucciones en la unidad de control

Al concluir estos cuatro pasos, en la unidad de control hay la instrucción ADD, y el contador del programa apunta al byte que contiene el número a sumar. Antes de analizar la forma en que se obedece la instrucción, veamos primero la *unidad de control*. Consta de un registro en el cual se coloca la instrucción actual, un decodificador que decodifica los 8 bits de que consta el byte de instrucción en 256 líneas separadas, una para cada posible valor que puede retener el byte. (Si no se utiliza la totalidad de los 256 posibles códigos para códigos de instrucción, la cantidad de líneas decodificadas puede ser menor.) Estas líneas decodificadas se amplian a la unidad de microprograma, junto con la salida decodificada desde un contador. En la unidad de microprograma se halla la verdadera función de cada código de instrucción, en forma de circuito lógico, y las salidas desde estos circuitos lógicos forman las diversas señales de habilitación de registros. Veamos ahora la parte "ejecutar" del ciclo:

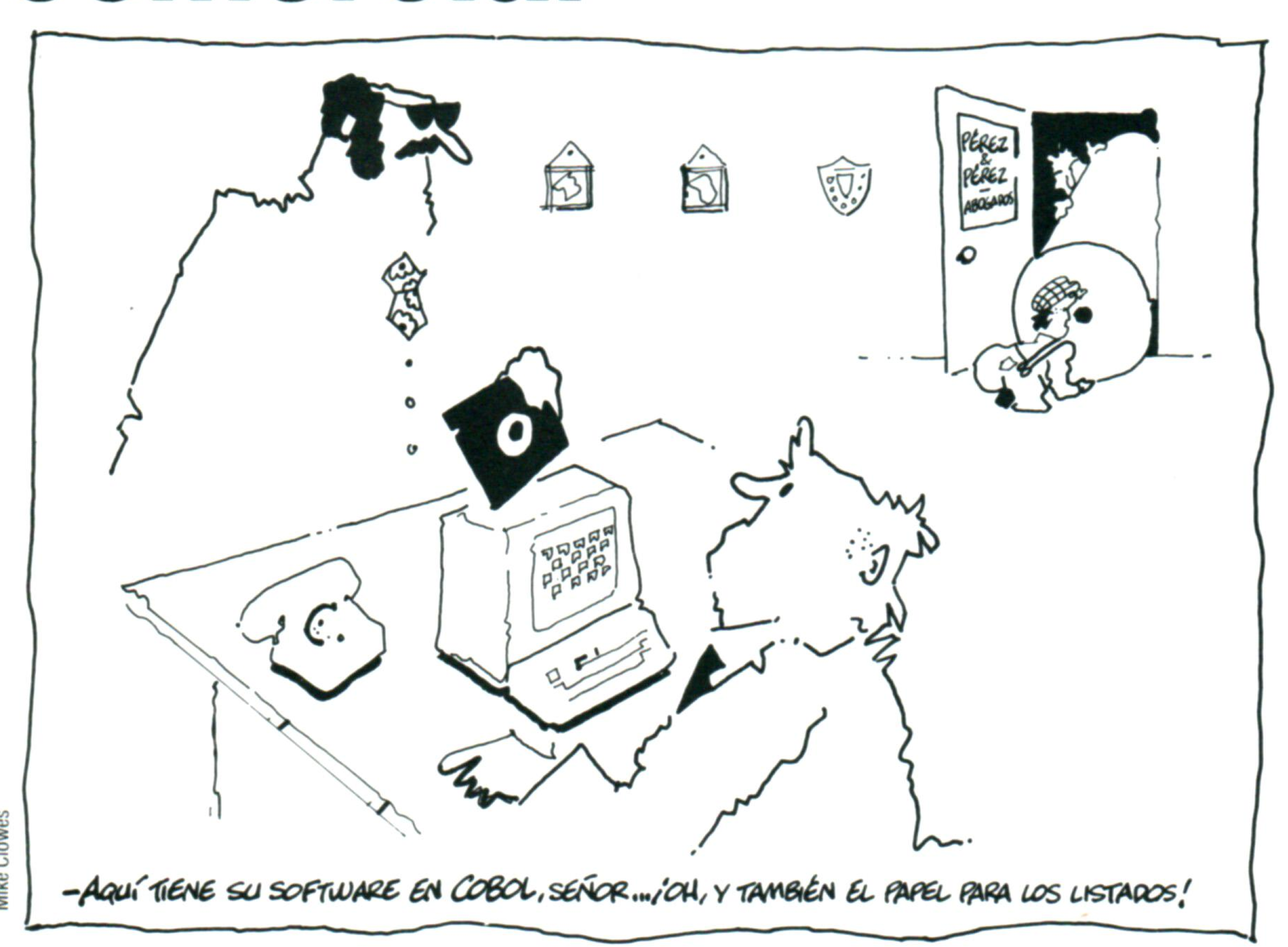
Paso	Señales control	Acción
5	EOA on PC EI on M	Traer el número a sumar a M
6	+1 on PC	Incrementar el PC listo para la sig. instrucción
7	EO on M EI on T	Copiar el núm. en reg. temporal ALU
8	selec. función ADD EO on T EO on A	Efectuar la suma
9	EO on ALU EI on A	Copiar el resultado en el acumulador

Mientras que la parte "buscar" del ciclo es la misma en todos los casos, la parte "ejecutar" está diseñada a medida para llevar a cabo una determinada tarea: en este caso, para tomar el siguiente byte del programa y sumárselo al valor en el acumulador. Cada código de instrucción posee su propio microprograma, implementado como una serie de puertas legicas. Si observamos una señal de habilitación determinada (la que habilita la salida desde el registro de datos de memoria [EO on M]), podemos ver cómo se implementa la anterior secuencia "ejecutar". Es necesario establecer EO on M en el paso 3 y el paso 7 del ciclo buscar-ejecutar. Supongamos que el código para nuestra instrucción ADD inmediata es 11001110, 206 en decimal. Simplemente aplicando impulsos de apertura a la línea C7 del contador de pasos con la línea 206 desde el decodificador de instrucciones y relacionando el resultado con C3 mediante una función OR, obtenemos una señal de habilitación de salida para el registro de datos de memoria. Por supuesto, muchas instrucciones diferentes podrían necesitar enviar señales de habilitación de salida a este registro, pero es tan sólo cuestión de relacionarlas entre sí mediante una OR.

Los fabricantes de procesadores utilizan máquinas especiales, denominadas ordenadores microprogramables, para programar los micropasos de una instrucción desde software. Éstas a menudo se utilizan para diseñar y desarrollar procesadores nuevos, en los que se puede desarrollar el conjunto de instrucciones antes de confiarlo al hardware en la unidad de control.



Líder comercial

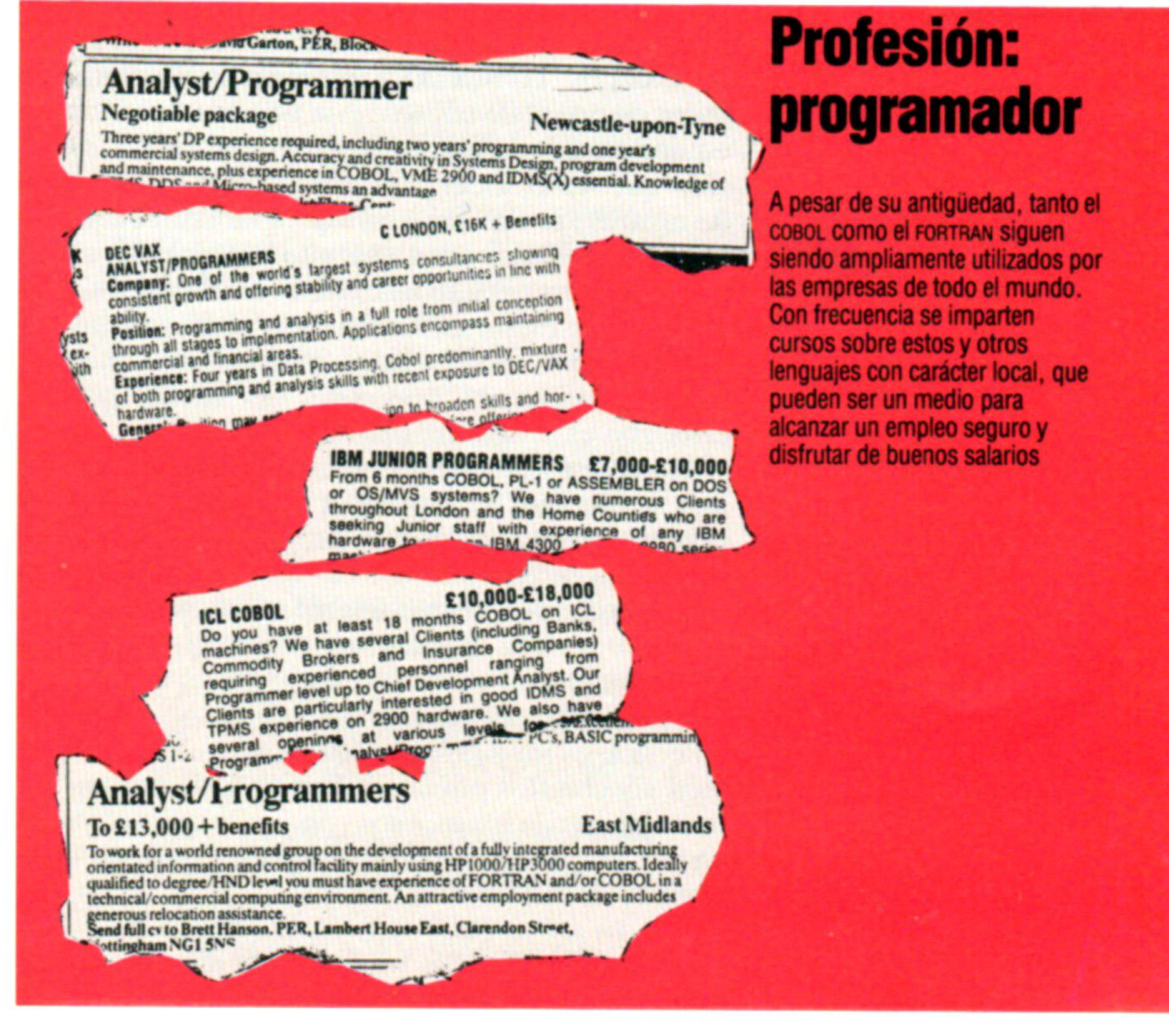


Mike Clowes

El COBOL es el lenguaje informático de uso más generalizado en las aplicaciones comerciales y de gestión

El COBOL (*Common Business Oriented Language*) es con mucho el lenguaje de programación más ambpliamente utilizado en todo el mundo, y si bien el BASIC tal vez sea el más conocido, es probable que las verdaderas líneas de código escrita sen COBOL superen a la de todos los otros lenguajes juntos. Esto no se debe sólo a que el COBOL sea prolijo, necesitando de muchas líneas aun para un programa simple, sino a que ha sido el principal lenguaje (casi el único) para uso comercial y de gestión.

El COBOL es un lenguaje muy estandarizado. De hecho, existe un cuerpo permanente, el comité CODASYL, que supervisa su uso y desarrollo. Este elevado nivel de normalización es importante por varios motivos. Es vital que una organización comercial sea capaz de transferir su software, lo que fácilmente podría ser uno de sus más valiosos bienes, de un ordenador a otro sin ningún cambio sustancial, o mejor aún, sin tener que introducir ningún cambio



Profesión: programador

A pesar de su antigüedad, tanto el COBOL como el FORTRAN siguen siendo ampliamente utilizados por las empresas de todo el mundo. Con frecuencia se imparten cursos sobre estos y otros lenguajes con carácter local, que pueden ser un medio para alcanzar un empleo seguro y disfrutar de buenos salarios



Estructura dividida
Los programadores familiarizados sólo con el BASIC es posible que opinen que la estructura de un programa en COBOL es un tanto extraña. No obstante, este lenguaje cumple a la perfección los requerimientos exigibles en cuanto a legibilidad, adaptabilidad y facilidad de revisión y actualización

en absoluto. Muchas empresas que operan con un gran ordenador central no desearán perder un valioso tiempo de esa máquina y esperarán que el desarrollo de programas se realice en una máquina completamente diferente, con lo que, nuevamente, resulta vital que los programas se puedan transferir con facilidad.

Otro factor es que los programas comerciales son bastante distintos de los programas en BASIC que hemos escrito en nuestros micros personales. Son muy largos, constando a menudo de decenas de miles de líneas de código, y se tienden a escribir en equipo en vez de por un único programador, situación que no es ideal para producir un código libre de errores. Se espera que los programas tengan una vida útil de varios años, durante los cuales se eliminarán los errores y se efectuarán correcciones, ampliaciones y mejoras. Todas estas revisiones las llevarán a cabo uno o más programadores, quienes, muy probablemente, no serán los mismos que escribieron el programa original.

Estas restricciones (y el hecho de que los programas se escriban de acuerdo a unas especificaciones rígidas) pueden hacer de la programación en COBOL una de las tareas más tediosas que existen. No obstante, existe muchísimo interés en el lenguaje en sí mismo y no hay ninguno mejor que él para tareas que requieran tratamiento de archivos complejos.

El aspecto y el funcionamiento de un programa en COBOL es muy diferente al de otros lenguajes, si bien encubiertos por su prolijidad podemos encontrar algunos de los mismos conceptos que subyacen en todos los lenguajes procedurales. Los programas constan de cuatro divisiones:

Identification division (división de identificación): es básicamente de índole documental, y proporciona un espacio para el nombre de un programa, el nombre del programador, fechas de escritura y compilación y comentarios introductorios.

Environment division (división de entorno): está concebida como la parte dependiente de la máquina y contiene detalles relativos al ordenador con el cual se escribió y ejecutó el programa. Muchos de los detalles que se incluyen aquí están ahora en manos de los sistemas operativos modernos, de modo que actualmente esta división se utiliza fundamentalmente para especificar archivos externos utilizados en el programa.

Data division (división de datos): contiene detalle sobre todas las zonas de datos utilizadas por el programa. En cierto sentido, es como la parte de declaraciones de un programa en PASCAL, donde se introducen los nombres de las variables, aunque es algo peligroso usar el concepto de "variable" en su acepción normal tratándose de datos de COBOL. En la división de datos hay dos secciones principales: la sección de archivos, que contiene esencialmente los trazados de registros para archivos externos, y la sección de trabajo-almacenamiento, para los datos utilizados sólo dentro de un programa.

Procedure division (división de procedimientos): aquí se especifican las operaciones y procedimientos a llevar a cabo sobre los datos de la división de datos. Si fuera necesario, la división de procedimientos se podría dividir en dos secciones, y cada sección o la división entera está compuesta de párrafos con nombre compuestos de oraciones. Una oración puede corresponder a una o más "sentencias" o "instrucciones" que a su vez se pueden subdividir en cláusulas para modificar o calificar la verdadera operación que se esté efectuando. La estructura de la división de procedimientos está claramente diseñada para parecerse lo más posible al inglés común, lo que, en unión con largos identificadores de hasta treinta caracteres, puede dar lugar a programas que resulten comprensibles no sólo a los programadores sino también a los no programadores. Sin embargo, escribir un "código espagueti" incomprensible en COBOL es tan fácil como en cualquier otro lenguaje.

Centraremos nuestra atención en la *data division* (división de datos), puesto que definir y designar las zonas de datos es una parte muy importante en un programa en COBOL. En relación a los datos, el punto de vista del COBOL consiste en considerar la división de datos como una larga serie de caracteres que se pueden dividir y subdividir a voluntad. Lo mejor es pensar en ella como si se tratara de un bloque de memoria. A las diversas divisiones que se efectúan en este bloque de datos se les asigna un nombre y se les otorga un número de nivel para indicar la estructura básica de los datos.

Los números de nivel normales van de 01 a 49, con algunos adicionales. Una sección de la división de datos se definirá en el nivel 01. Esta sección se puede luego subdividir otorgándoles a las subsecciones números de nivel superior (02, p. ej.), si bien los números no necesitan ser consecutivos. Una de estas subsecciones se puede volver a subdividir mediante el empleo de un número de nivel aún mayor, y así sucesivamente.

Los nombres de datos pueden tener hasta 30 caracteres de longitud, usando solamente caracteres alfabéticos y numéricos y guiones. El COBOL posee



un gran vocabulario de palabras reservadas que no se pueden utilizar para nombres de datos. No obstante, son pocas las que poseen guiones, de modo que no es difícil superar este problema.

Es necesario distinguir entre dos tipos de datos: elementales y grupales. Un dato elemental es aquel que no está subdividido; en la división de procedimientos se puede aludir tanto a datos elementales como a grupales. Cada dato elemental ha de tener un USAGE y un PICTURE. Hay dos USAGE principales: COMPUTACION y DISPLAY.

Como puede suponer, COMPUTATIONAL se utiliza para datos numéricos destinadosa cálculos y hará que el número se almacene en forma binaria en vez de una serie de caracteres, realizando, en consecuencia, los cálculos aritméticos con mayor rapidez. No obstante, un dato no tiene que ser COMPUTATIONAL para poderse utilizar para cálculos. DISPLAY es el empleo pordefecto normal, que simplemente significa que el dato se alamacena como una serie de caracteres. Cada uno de estos dos USAGEs puede ser objeto de otros refinamientos, según el sistema que se utilice; por ejemplo, COMPUTATIONAL-3 para un número de punto flotante. La cláusula USAGE es opcional; si se omite, entonces se da por sentado DISPLAY.

La cláusula PICTURE define el contenido esperado de cada posición de carácter en un dato elemental. Los principales símbolos utilizados con: 9 para indicar un carácter numérico; A para un carácter alfabético; X para un carácter alfanumérico; y V para indicar la posición de un punto decimal en un dato numérico. Normalmente el punto decimal no se almacena, pero el COBOL lleva el registro de su posición para que los cálculos se puedan realizar correctamente.

Una S indica que un dato numérico llega signo. Si se omite, entonces se toma automáticamente el valor absoluto de cualquier número almacenado. Hay opciones en cuanto a almacenar el signo como un carácter separado o no. (Observe que PIC es la forma abreviada de PICTURE, y que se puede utilizar cualquiera de las dos.) Por último, además de un PICTURE y un USAGE, a cualquier dato se le puede asignar un valor inicial o especificarlo como una matriz.

Al igual que el FORTRAN, el COBOL se diseñó originalmente para usar con tarjetas perforadas. La longitud de cada línea no puede ser mayor de 72 caracteres y, de ser necesario, se debe utilizar una línea de continuación. El trazado está especificado estrictamente, pero muchos sistemas permitirán un trazado más libre si el programa no está destinado para otra máquina. Las seis primeras columnas se utilizan para números de línea. El COBOL no requiere números de línea, pero se pueden incluir para ayudar a la depuración. La columna 7 se usa para marcar una línea de continuación; un asterisco en esta posición indica un comentario. Las columnas de las 8 a la 11 son la zona A. Aquí empiezan los nombres de división, sección y párrafos, así como los datos de nivel 01. Otros datos y sentencias de la división de procedimientosempiezan en la zona B: columnas de la 12 a la 72. La mayoría de las declaraciones de datos constituyen una oración y, por tanto, deben terminar con un punto. Si en un dato se necesitan caracteres a los que realmente no se alude en la división de procedimientos, entonces se utiliza la palabra reservada FILLER.

Detalles personales

Definición de datos para un registro de un archivo de personal:

```
01 EMPLEADO-REGISTRO.
   02 EMPLEADO-NOMBRE.
      03 EMPLEADO-INITIAL PIC A.
      03 FILLER PIC X.
      03 EMPLEADO-APELLIDO PIC X(15).
   02 EMPLEADO-DEPARTAMENTO PIC XXX.
   02 EMPLEADO-SALARIO-CATEGORIA PIC 9.
   02 EMPLEADO-FECHA-NACIMIENTO.
      03 DIA-NACIMIENTO PIC 99.
      03 MES-NACIMIENTO PIC 99.
      03 AÑO-NACIMIENTO PIC 99.
```

Registro de personal
Vemos los tipos de datos utilizados para mantener un registro de información personal sobre los empleados. Observe el uso de PIC para especificar los tipos de datos esperados: 9 indica un número, A un caracter alfabético y X un carácter alfanumérico

Definición de datos para una línea de cabecera (*heading*) destinada a producir un informe a partir del archivo anterior:

```
s  01HEADING-LINE.
   02 HEADING-1 PIC X(4) VALUE "NOMBRE".
   02 FILLER PIC X(20)VALUE SPACES.
   02 HEADING-2 PIC X(10) VALUE
      "DEPARTAMENTO".
   02 FILLER PIC X(3) VALUE SPACES.
   02 HEADING-3 PICX(9) VALUE "SAL
      CATEG".
   02 FILLER PIC X(10) VALUE SPACES.
   02 HEADING-4 PIC X(3) VALUE "FECHA
      NACIM".
```

Diseño del trazado
Aquí se define el trazado del informe a generar. A los datos se les pueden asignar valores iniciales. HEADING-2, por ejemplo, consta de 10 caracteres alfanuméricos y se le ha asignado el valor DEPARTAMENTO

Definición de datos para una línea de salida destinada a producir un informe a partir del archivo de personal:

```
01OUTPUT-LINE.
   02 OUTPUT-NOMBRE.
      03 OUTPUT-INITIAL PIC A.
      03 FILLER PIC X VALUE".".
      03 OUTPUT-APELLIDO PIC X(15).
   02 FILLER PIC X(7) VALUE SPACES.
   02 OUTPUT-DEPARTAMENTO PIC XXX.
   02 FILLER PIC X(10) VALUE SAPCES.
   02 OUTPUT-SALARIO-CATEGORIA PIC 9.
   02 FILLER PIC X(18) VALUE SPACES.
   02 OUTPUT-FECHA-NACIMIENTO.
      03 OUTPUT-DIA PIC 99.
      03 FILLER PIC X VALUE"/".
      03 OUTPUT;MES PIC 99.
      03 FILLER PIC X VALUE"/".
      03 OUTPUT-AÑO PIC 99.
```

Rellenando espacio
Estas líneas especifican la línea de salida desde el archivo de personal. En los datos se pueden incluir caracteres aun cuando no se aluda a ellos en la división de procedimientos; esto se hace utilizando la palabra reservada FILLER, que se usa aquí y en el listado anterior para insertar espacios

Definición de datos para una matriz unidimensional de veinte números de 3 dígitos y una matriz unidimensional de 10 * 20 números de 3 dígitos (con signo y un lugar decimal):

```
01 DATA-TABLE-1.
   02 DATA-TABLE-ENTRY PIC S99V9 USAGE
      COMPUTATIONAL OCCURS 20 TIMES.
01 DATA-TABLE-2.
   02 DATA-TABLE-ROW OCCURS 10 TIMES.
      03 DATA-TABLE-ENTRY-2 PIC S99V9
         USAGE
         COMPUTATIONAL OCCURS 20 TIMES.
```

Hecho para ser veloz
Declarar un dato como COMPUTATIONAL, como hace esta sección del programa, significa que los valores se almacenan en formato binario en lugar de en caracteres. Ello hace que los ulteriores cálculos que utilicen el dato sean mucho más rápidos. No obstante, los datos no han de definirse necesariamente de esta manera si usted quiere efectuar operaciones aritméticas con ellos, pero los cálculos son datos retenidos como caracteres serán más lentos

Facilidades adicionales

Añadiremos algunas nuevas funciones a nuestro programa de hoja electrónica

Cuando se prepara una hoja electrónica para realizar una serie de cálculos, con frecuencia las fórmulas a entrar en cada celda de una columna o fila determinadas son similares. Por ejemplo, quizá quisiéramos sumar cada elemento de la primera columna con cada elemento de la segunda columna y colocar los resultados en la tercera. para llevar a cabo esta sencilla tarea, sería necesario entrar una fórmula en cada celda de la tercera columna. Sería preciso programar la primera celda de esta columna, A3, con fórmula A1 + A2; sería necesario programar la siguiente celda de la tercera columna, B3, con la fórmula B1 + B2, y así sucesivamente. Si bien las fórmulas no son idénticas, son similares, de modo que para evitar la tediosa tarea de tener que entrar manualmente las fórmulas en todas las celdas de la tercera columna, sería de gran utilidad contar con una facilidad que entrara automáticamente fórmulas adecuadamente modificadas en una fila o columna entera (o en parte de ella). Este cometido lo lleva a cabo la función Reproducir (*replicate*).

Las funciones Borrar (*clear*), Almacenar (*store*) y Recuperar (*retrieve*) actúan sobre la totalidad de la hoja y, como sus nombres indican, permiten que el usuario borre los datos de las celdas de la hoja completa, almacene los datos para toda la hoja en la memoria (en lugar de en disco o cinta) y los recupere ulteriormente.

Otra función útil para moverse por la hoja es la función Tab. Si bien es posible trasladarse a cualquier parte de la hoja mediante el cursor, ello puede ser bastante lento, en especial porque la hoja se ha de reescribir cada vez que la ventana de pantalla desfila por la hoja. La función Tab permite que el usuario salte hasta cualquier celda simplemente entrando el nombre de la celda.

La función "Reproducir"

Las subrutinas que manipulan la función Reproducir están ubicadas entre las líneas 5400 y 5995. La primera rutina toma la fórmula a reproducir y la descompone en los elementos que la constituyen. Esta rutina en realidad es llamada por la rutina Reproducir principal y trabaja con una fórmula tomada de la matriz de fórmulas de infijos F$() y almacenada en C$. La fórmula se descompone en sus operadores, nombre de celdas y constantes, y se coloca en la matriz E$(). La rutina va explorando C$ carácter por carácter, y si el actual no es un operador se le añade una serie temporal, T$. Si el carácter *es* un operador, entonces se coloca T$ en el siguiente elemento libre de E$() (ya que encontrar un operador significa que en T$ hay almacenado un operando completo). El operador que se acaba de encontrar también se añade a E$(). El proceso continúa hasta haber explorado la fórmula en su totalidad.

La siguiente sección de código, entre las líneas 5500 y 5650, contiene la rutina "entrar y verificar". Ésta permite que el usuario dirija la función Reproducir para que actúe sobre un grupo de celdas determinado y está escrita para aceptar instrucciones con el formato A1(B1-F1. Esto significa: tomar la fórmula de la celda A1 y reproducirla en la columna 1 entre B1 y F1. Por el contrario, para reproducir a lo largo de una fila, se podría entrar una instrucción como ésta: A1(A2-A10. La primera sección de la rutina de entrada, entre las líneas 5500 y 5520, comprueba que la entrada posea una sintaxis correcta. A continuación, se divide la entrada para extraer los tres nombres de celda sobre las que es necesario trabajar. Estos nombres se colocan en seguida en R1$, R2$ y R3$.

La siguiente sección, en la línea 5700, es el verdadero punto de entrada a la rutina Reproducir, siendo su función la de decidir si la instrucción implica que se ha de reproducir una fila o una columna y saltar a la rutina para reproducir una fila o una columna adecuada. Se puede decidir si se ha de reproducir una fila o una columna examinando los tres nombres de celdas entrados como parte de la instrucción Reproducir. Cada nombre de celda se compone de una letra seguida por un número. Si cada uno de los tres nombres de celda comienzan con la misma letra, como en A1(A2-A10, luego está claro que la reproducción de fórmula debe realizarse a lo largo de una fila, en este caso, la fila A. Sin embargo, si la parte numérica de cada nombre de celda es la misma, por ejemplo A3(B3-H3, entonces se ha de reproducir una columna. Un efecto colateral de la comprobación de fila o de columna es que las entradas ilegales, tales como A1(B1-F2, se detectan automáticamente.

En las líneas 5800 a 5895 encontramos el código que reproduce una fórmula a lo largo de una columna. En primer lugar, se llama a la rutina que analizábamos primero y que descompone la fórmula a reproducir. En la línea 5820, un bucle FOR...NEXT utiliza los valores ASCII de los primeros caracteres de R2$ y R3$ (los nombres de las celdas entre las cuales se ha de reproducir la fórmula) como sus variables de control. Entonces comienza, en la línea 5830, la tarea de construir la nueva fórmula para cada celda.

En esta etapa es interesante analizar las limitaciones de la función Reproducir. Cuando usted reproduce una fórmula a lo largo de una columna, la función sólo trabajará adecuadamente si cada nombre de celda dentro de la fórmula a reproducir posee la misma letra de fila. Ello significa que una fórmula como A1+A2*A3 se reproducirá con todo éxito, no así se tratara de A1+A2*B2. Sin embargo, ésta no es una restricción seria, porque la mayoría de las aplicaciones que se valen de la función Reproducir emplean fórmulas que trabajan sobre elementos de una sola fila.

"Borrar", "Almacenar", "Recuperar" y "Tab"

El código para todas estas cortas rutinas se da entre las líneas 5000 y 5395. Borrar la hoja es una cues-

tión muy simple. La matriz (M(,), utilizada para retener los datos de celda para toda la hoja, se establece en cero mediante un par de bucles FOR...NEXT anidados, y en virtud de la llamada a la subrutina de la línea 1700 se reimprime la hoja de la pantalla.

Para almacenar y recuperar la hoja de datos se emplea una segunda matriz, N(,). Las rutinas almacenar y recuperar transfieren el contenido de M(,) a N(,), o viceversa.

La rutina Tab empieza en la línea 5200 y acepta como entrada un nombre de celda. Antes de reescribir la hoja, es necesario volver a calcular los límites horizontales y verticales de la ventana de pantalla. Además de reimprimir los datos de la hoja, se han de imprimir nuevas etiquetas de columna y fila para dar cabida al hecho de que la ventana de la hojua puede haberse modificado a consecuencia del desplazamiento a una nueva celda.

Complementos al BASIC

BBC Micro:

Introduzca estos cambios en la versión para el C64 (asegúrese de que las sentencias PRINT de las líneas 5220 y 5502 contengan espacios suficientes para borrar una línea completa):

```
5210 PRINT TAB(0,22);
5220 PRINT "NUEVA
     CELDA:                    "
5501 PRINT TAB(0,22;
5502 PRINT
     "REPRODUCIR:              "
```

Amstrad CPC 464/664:

Introduzca estos cambios en la versión para el C64 (asegúrese de que las sentencias PRINT de las líneas 5220 y 5502 contengan espacio suficiente para borrar una línea entera)

```
5210 LOCATE 1,22
5220 PRINT "NUEVA
     CELDA:                    "
5501 LOCATE 1,22
5502 PRINT
     "REPRODUCIR:              "
```

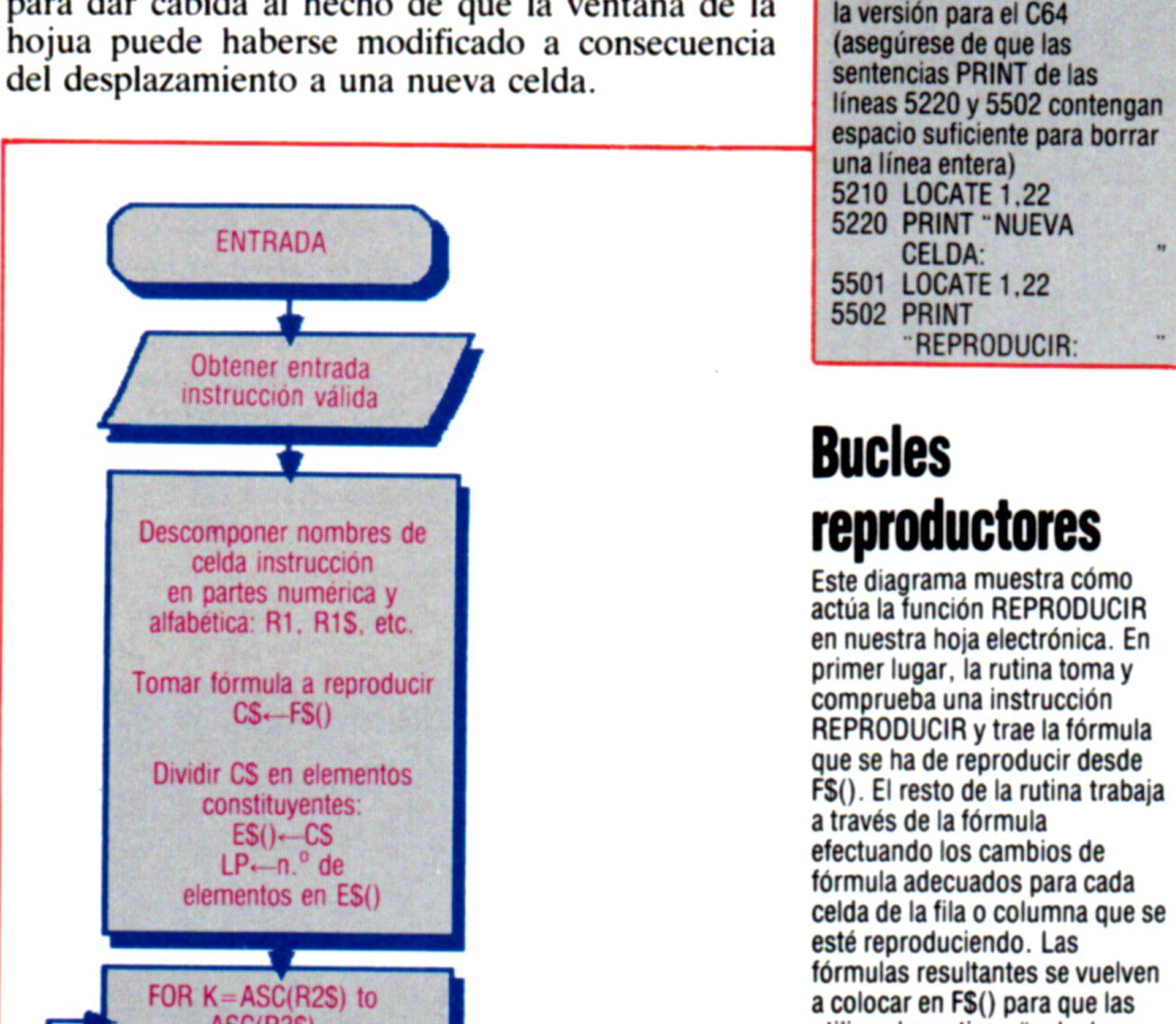
Facilidades adicionales

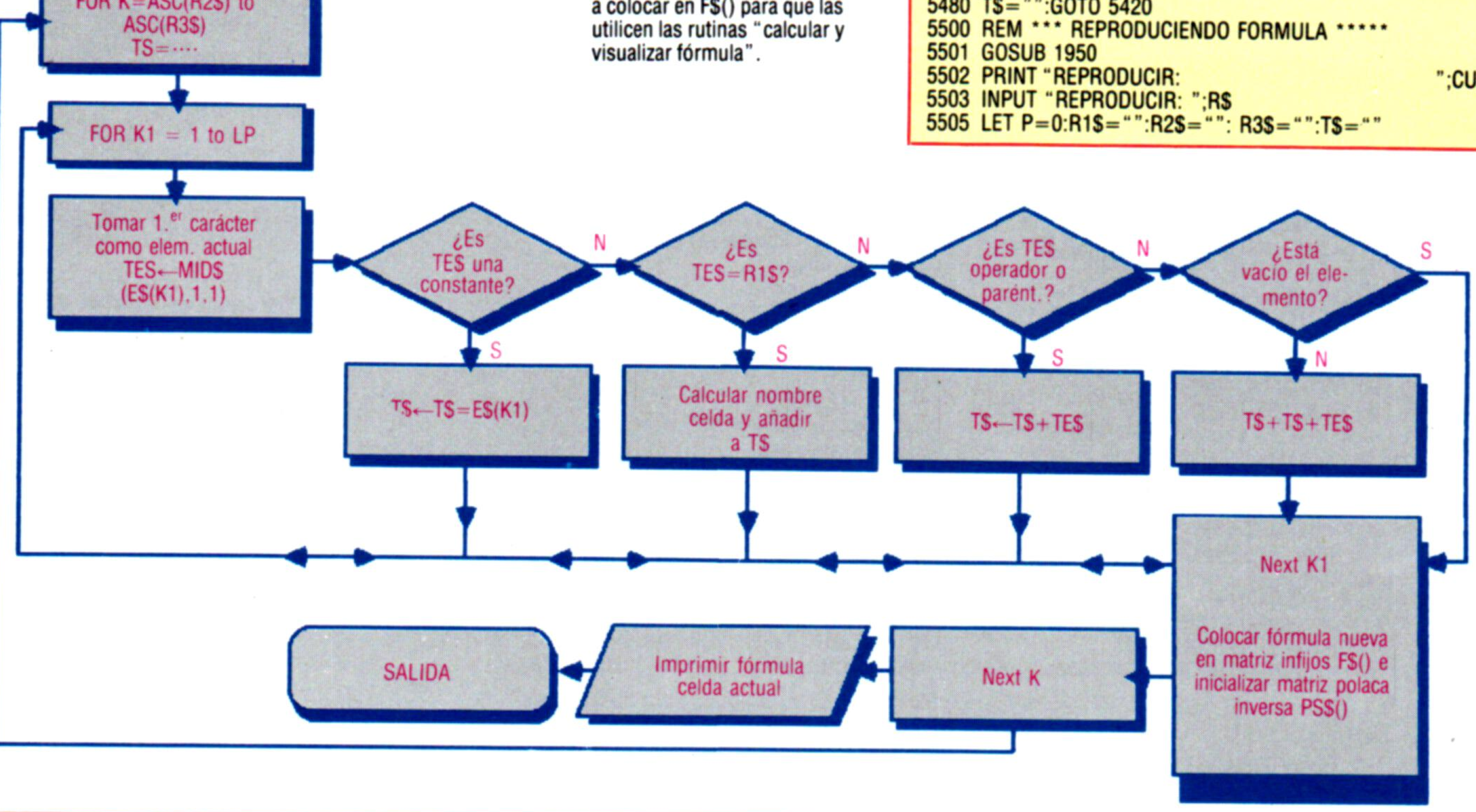
Commodore 64:

```
2320 >LET P$=H$((J-1)*)15+ I,1TO ):LET
     I$=F$((J-1)*15+I,1TO)
5000 REM **** BORRAR MATRIZ ****
5010 FOR I=1 TO 15:FOR J=1 TO 15:M(I,J)=0:NEXT J,I:
     GOSUB 1700:RETURN
5100 REM **** TOMAR HOJA ANTERIOR ****
5110 FOR I=1 TO 15:FOR J=1 TO 15
5120 M(I,J)= N (I,J):NEXT J,I
5130 GOSUB 1700:RETURN REM IMPRIMIR DATOS
5150 REM ***** ALMACENAR HOJA ACTUAL ****
5160 FOR I=1 TO 15:FOR J=1 TO 15
5170 N(I,J)=M(I,J):NEXT J,I
5180 GOSUB 1700:RETURN:REM IMPRIMIR DATOS
5200 REM **** RUTINA GOTO CELDA ****
5210 GOSUB 1950:REM MOVER HASTA LINEA ENTRADA
5220 PRINT "NUEVA CELDA:                            ";CU$
5230 INPUT "NUEVA CELDA:",NC$
5240 LET NC$=MID$(NC$,1,3)
5250 LET N1$=MID$(NC$,1,1):LET N2$=MID$(NC$,2,2)
5260 IF N1$< "A" OR N1$> "0" THEN 5210
5270 IF VAL(N2$)<1 OR VAL(N2$)> 15 THEN 5210
5280 GOSUB 1600:REM APAGA CURSOR
5300 LET Y=AS(N1$)-64
5305 LET X=VAL(N2$)
5315 IF X< 6 THEN H1=1:H2=H1+4:GOTO 5330
5316 IF X<11 THEN H1=6:H2=H1+4:GOTO 5330
5320 H1=11:H2=15
5330 IF Y<9 THEN V1=1:V2=V1+7:GOTO 5360
5340 V1=8:V2=V1+7
5360 GOSUB 1800:REM IMPRIMIR COLUMNAS
5370 GOSUB 1850:REM IMPRIMIR FILAS
5390 GOSUB 1700:REM REIMPRIMIR HOJA
5395 GOTO 1100
5400 REM **** CONFECCIONAR LISTA DE ELEMENTOS DE C$
     ****
5405 P=0:LP=1:T$="":TE$""
5410 FOR K=1 TO 20:E$(K)="":NEXT K
5420 LET P=P+1:IF P> LEN(C$) THEN E$(LP)=T$:RETURN
5430 TE$=MID$(C$,P,1)
5440 IF TE$="+" OR TE$="-" OR TE$="*"OR TE$="/"
     THEN 5460
5445 IF TE$="^" OR TE$="(" OR TE$=")" THEN 5460
5450 T$=T$+ TE$:GOTO 5420
5460 IF T$="" THEN 5470
5465 E$(LP)=T$:LP=LP+1
5470 E$(LP)=TE$:LP=LP+1
5480 T$="":GOTO 5420
5500 REM *** REPRODUCIENDO FORMULA *****
5501 GOSUB 1950
5502 PRINT "REPRODUCIR:                             ";CU$
5503 INPUT "REPRODUCIR: ";R$
5505 LET P=0:R1$="":R2$="": R3$="":T$=""
```

Bucles reproductores

Este diagrama muestra cómo actúa la función REPRODUCIR en nuestra hoja electrónica. En primer lugar, la rutina toma y comprueba una instrucción REPRODUCIR y trae la fórmula que se ha de reproducir desde F$(). El resto de la rutina trabaja a través de la fórmula efectuando los cambios de fórmula adecuados para cada celda de la fila o columna que se esté reproduciendo. Las fórmulas resultantes se vuelven a colocar en F$() para que las utilicen las rutinas "calcular y visualizar fórmula".





Caroline Clayton

```
5510 IF MID$(R$,3,1)<>"(" AND MID$(R$,4,1)<>"(" THEN
     5500
5515 IF MID$(R$,6,1)<> "-" AND MID$(R$,7,1)<> "-" AND
     MID$(R$,8,1)<> "-" THEN 5500
5517 IF R$="" THEN 5500
5520 LET P=P+1
5530 LET T$=MID$(R$,P,1)
5540 IF T$"(" THEN P=P+1:GOTO 5570
5550 LET R1$=R1S+T$
5560 GOTO 5520
5570 LET T$=MID$(R$,P,1)
5580 IF T$="-" THEN P=P+1:GOTO 5610
5590 LET R2$=R2$+T$
5600 LET P=P+1:GOTO 5570
5610 LET T$=MID$(R$,P,1)
5620 LET R3$=R3$+T$
5630 LET P=P+1
5640 IF P<= LEN (R$) THEN 5610
5650 RETURN
5700 REM ** DECIDIR COLUMNA O FILA ***
5730 GOSUB 5500:REM COMPROBAR SI ENTRADA
     VALIDA
5765 R1=VAL(MID$(R1$,2)):R1$=MID$(R1$,1,1)
5770 R2=VAL(MID$(R2$,2)):R2$=MID$(R2$,1,1)
5775 R3=VAL(MID$(R3$,2)):R3$=MID$(R3$,1,1)
5780 IF R1=R2 AND R1=R3 THEN 5800
5790 IF MID$(R1$,1,1)=MID$(R2$,1,1)
     AND MID$(R1$,1,1)=MID$(R3$,1,1) THEN 5900
5795 GOTO 5700
5800 REM **** REPRODUCIR COLUMNA ****
5805 LET C$=F$((ASC(R1$)-65)* 15+R1)
5810 GOSUB 5400:REM CONFECCIONAR LISTA DE
     ELEMENTOS
5820 FOR K=ASC (R2$) TO ASC(R3$):T$=""
5830 FOR K1=1 TO LP
5840 LET TE$=MID$(E$(K1),1,1)
5845 IF TE$> ="0" AND TE$< ="9" OR TE$="." THEN
     T$=T$+E$(K1):GOTO 5880
5850 IF TE$=R1$ THEN T$=T$+ CHR$(K) +
     MID$(E$(K1),2):GOTO 5880
5860 IF  TE$="+"  OR  TE$="-"  OR  TE$="*"  THEN
     T$=T$+TE$:GOTO 5880
5865 IF TE$="/" OR TE$="^" OR TE$="(" OR TE$=")" THEN
     T$=T$+ TE$:GOTO 5880
5870 IF TE$<>"" THEN T$=T$+ TE$
5880 NEXT K1
5890 F$((K-65)*15+R1)=T$:PS$((K-65)*15+R1)=""
5895 NEXT K:GOSUB 1900:RETURN
5900 REM **** REPRODUCIR FILA ****
5905 LET C$=F$((ASC(R1$)-65)*15+R1)
5910 GOSUB 5400:REM CONFECCIONAR LISTA DE
     ELEMENTOS
5920 FOR K=R2 TO R3:T$=""
5930 FOR K1=1 TO LP
5940 LET TE$=MID$(E$(K1),1,1)
5945 IF (TE$> ="0" AND TE$ <="9") OR TE$="." THEN
     T$=T$+E$(K1):GOTO 5980
5950 IF TE$>="A" AND TE$<="0"
     THEN T$=T$+T$+MID$(STR$(K),2):GOTO 5980
5960 IF  TE$="+"  OR  TE$="-"  OR  TE$="*"  THEN
     T$=T$+TE$:GOTO 5980
5965 IF TE$="/" OR TE$="^" OR TE$="(" OR TE$=")" THEN
     T$=T$+TE$:GOTO 5980
5970 IF TE$<>"" THEN T$=T$+TE$
5980 NEXT K1
5990 F$((ASC(R1$)-65)*15+K)=T$:PS$((ASC(R1$)-65
     *15+K)=""
5995 NEXT K: GOSUB 1900:RETURN
```

Spectrum:

```
5000 >REM *** BORRAR MATRIZ ***
5010 DIM M(15,15):GO SUB 1700: RETURN
5100 REM *** TOMAR HOJA ANTERIOR ****
5110 FOR I=1 TO 15: FOR J=1 TO 15
5120 LET M(I,J)=N(I,J)
5130 NEXT J: NEXT I
5140 GO SUB 1700: RETURN
5150 REM ALMACENAR HOJA ACTUAL EN
     MEMORIA
5160 FOR I=1 TO 15: FOR J=1 TO 15
5170 LET N(I,J)=M(I,J): NEXT J: NEXT I
5180 GO SUB 1700: RETURN
5200 REM **** RUTINA GOTO CELDA ****
5210 PRINT AT 18.0;"          ENTRE NUEVA
     CELDA"
5220 INPUT LINE N$
5225 IF LEN (N$) < 3 THEN LET N$=N$+ " "
5230 LET N$=N$(1 TO 3):LET O$=N$(2 TO 3)
5240 LET N$=N$(1)
5250 IF N$<"A" OR N$> "0" THEN GO TO 5210
5260 IF VAL (O$)<1 OR VAL (O$)> 15 THEN GO TO
     5210
5280 GO SUB 1600: REM APAGAR CURSOR
5300 LET Y=CODE (N$)-64
5305 LET X=VAL (O$)
5310 LET VL=V2: LET HL=H2
5315 IF X< H1 THEN GO TO 5330
5316 IF X< H2 THEN GO TO 5335
5320 IF X>=12 THEN LET H1=12:LET H2=15: GO
     TO 5340
5330 LET H1=X: LET H2=H1+3
5335 IF Y< V1 THEN GO TO 5350
5336 IF Y< V2 THEN GO TO 5360
5340 IF Y> =9 THEN LET V1=9: LET V2=
     V1+6: GO TO 5360
5350 LET V1=Y: LET V2=Y+6
5360 GO SUB 1800: REM IMPRIMIR COLUMNAS
5370 GO SUB 1850: REM IMPRIMIR FILAS
5380 IF V2=VL AND H2=HL THEN GO SUB 1650:
     GO TO 1100
5390 GO SUB 1700
5395 GO TO 1100
5400 REM **** CONFECCIONAR LISTA DE
     ELEMENTOS DE I$ ***
5405 LET P=0: LET LP=1: LET T$="":LET U$=""
5410 DIM E$(20,5)
5420 LET P=P+1: IF P> LEN (C$) THEN LET
     E$(LP)=T$: RETURN
5430 LET U$=C$(P)
5440 IF U$="+" OR U$="-" OR U$="*" OR
     U$="/" THEN GO TO 5460
5445 IF U$="^" OR U$="(" OR U$=")" THEN GO
     TO 5460
5450 LET T$=T$+U$: GO TO 5420
5460 LET E$(LP)=T$: LET LP=LP+1
5470 LET E$(LP)=U$: LET LP=LP+1
5480 LET T$="": GO TO 5420
5500 REM *** REPRODUCIENDO ***
5505 LET P=0: LET X$="": LET Y$"": LET Z$="":
     LET T$=""
5510 IF R$(3)<>"(" AND R$(4)<>"(" THEN GO
     TO 5660
5515 IF R$(6)<> "-" AND R$(7)<>"-" AND
     R$(8)<>"-" THEN GO TO 5660
5520 LET P=P+1
5530 LET T$=R$(P)
5540 IF T$="("THEN LET P=P+1:GO TO
     5570
5550 LET X$=X$+T$
5560 GO TO 5520
5570 LET T$=R$(P)
5580 IF T$="-" THEN LET P=P+1:GO TO
     5610
5590 LET Y$=Y$+T$
5600 LET P=P+1: GO TO 5570
5610 LET T$=R$(P)
5620 LET Z$=Z$+T$
5630 LET P=P+1
5640 IF P< = LEN (R$) THEN GO TO 5610
5650 RETURN
5670 PRINT AT 18,0;"ERROR EN SERIE
     REPRODUCIR"
5680 RETURN
5700 REM **** DECIDIR COLUMNA O FILA ****
5710 PRINT AT 18,0;"REPRODUCIR:"
5720 INPUT LINE R$
5730 GO SUB 5500
5765 LET R1=VAL (X$(2TO )): LET X$=X$(1)
5770 LET R2=VAL (Y$(2 TO )): KET Y$=Y$(1)
5775 LET R3=VAL (Z$(2 TO )): LET Z$=Z$(1)
5780 IF R1=R2 AND R1=R3 THEN GO TO 5800
5790 IF X$=Y$ AND X$=Z$ THEN GO TO 5900
5795 GO TO 5700
5800 REM **** REPRODUCIR COLUMNA *****
5805 LET C$=F$((CODE (X$)-65)*15+R1)
5810 GO SUB 5400: REM ELABORAR LISTA DE
     ELEMENTOS
5820 FOR K=CODE (Y$) TO CODE (Z$): LET T$=""
5830 FOR J= TO LP
5840 LET U$=E$(J) (TO 1)
5850 IF U$=X$ THEN LET T$=T$+ CHR$ (K)+
     E$(J,2 TO ): GO TO 5800
5860 IF U$="+ " OR U$="-" OR U$="*" THEN
     LET T$=T$ +U$: GO TO 588
5865 IF U$="/" OR U$="^" OR U$="(" OR
     U$=")" THEN LET T$=T$+U$: GO TO 5880
5870 IF U$<>"" THEN LET T$=T$+U$
5880 NEXT J
5882 LET U$="": FOR i=1 TO LEN (T$): IF T$ (i TO
     i)<>" " THEN LET U$= U$+T$(i TO i)
5885 NEXT i
5890 LET F$((K-65)*15+R1)=U$: LET
     H$((K-65)* 15+R1)=""
5895 NEXT K: GO SUB 1900: RETURN
5900 REM **** REPRODUCIR FILA ****
5905 LET C$=F$((CODE (X$)-65)*15+R1)
5910 GO SUB 5400: REM CONFECCIONAR LISTA DE
     ELEMENTOS
5920 FOR K=R2 TO R3: LET T$=""
5930 FOR J=1 TO LP
5940 LET U$=E$(J,1 TO 1)
5950 IF U$> ="A" AND U$< = "0" THEN LET
     T$=T$+U$+STR$ (K): GO TO 5980
5960 IF U$="/" OR U$="^" OR U$="(" OR
     U$=")" THEN LET T$=T$+U$: GO TO 5980
5970 IF U$<>"" THEN LET T$=T$+U$
5980 NEXT J
5990 LET F$((CODE(X$)-65)*15+K)=T$: LET
     H$((CODE (X$)-65)*15+K)=""
5995 NEXT K: GO SUB 1900: RETURN
```

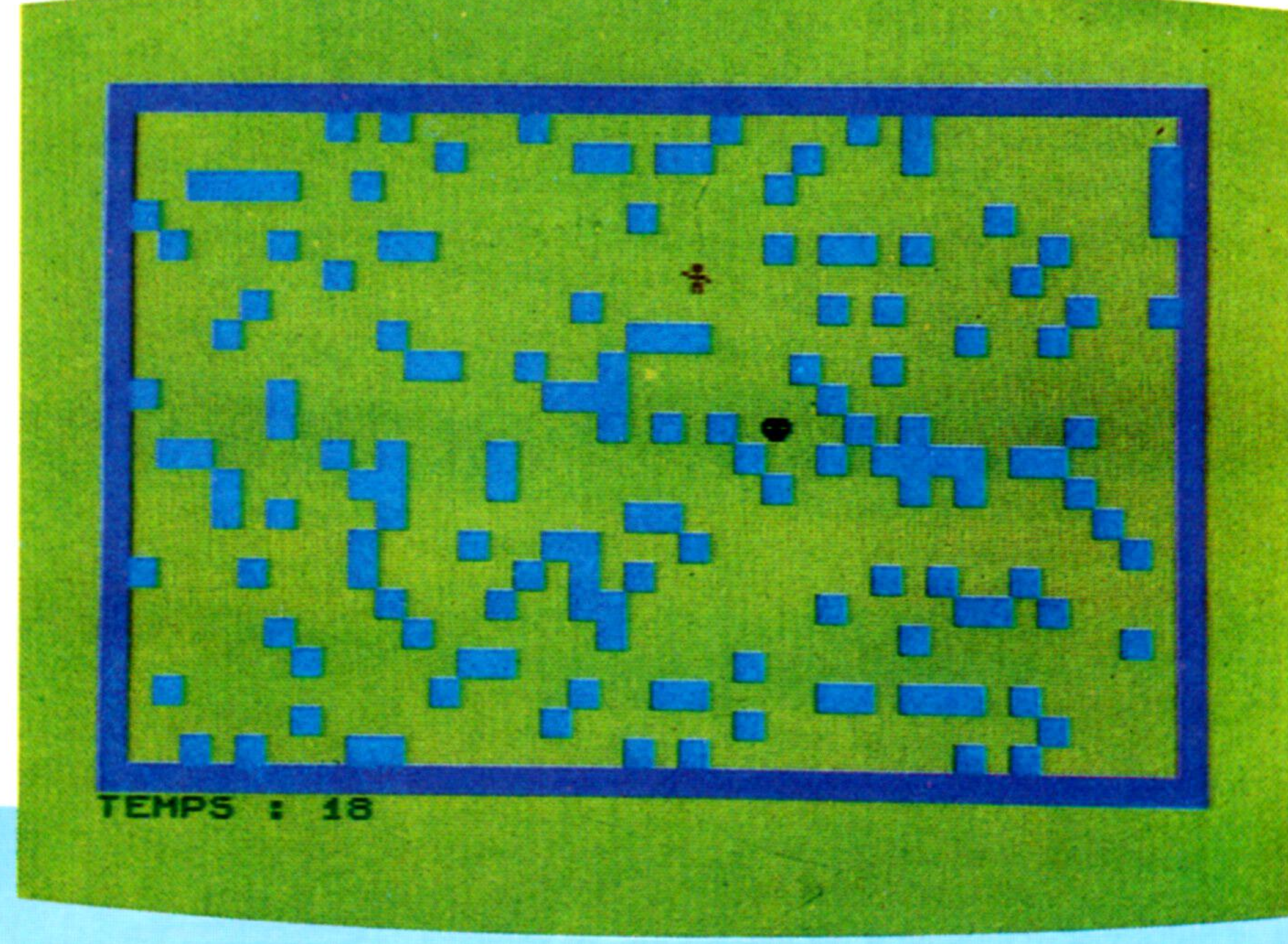


Persecución

El ladrón, representado por una máscara negra, ha escapado llevándose el botín... He aquí otra vez este conocido tema de la informática recreativa escrito en BASIC para el MO5 de Thomson

Se esconde en la ciudad, y usted tiene treinta minutos para encontrarlo y detenerlo. Atención, ¡no se precipite! Efectivamente, si se echa sobre él sin pensar, tiene todas las posibilidades de escapársele. La mejor manera de cogerlo es alcanzarlo de lado. (Es el método más eficaz a condición de no fallar). Si no se siente lo bastante seguro de sí mismo, atáquelo de frente, lo cual es más fácil pero mucho menos eficaz, ya que no es tan discreto. *Otro consejo*: no intente perseguirlo; esto no le daría resultado, pues él es mucho más rápido que usted. Ha de observar sus movimientos, como un detective. Cuando vea que da la vuelta, acérquese sin hacer ruido y sorpréndalo en el momento justo. Pero recuerde: ¡el tiempo va pasando!

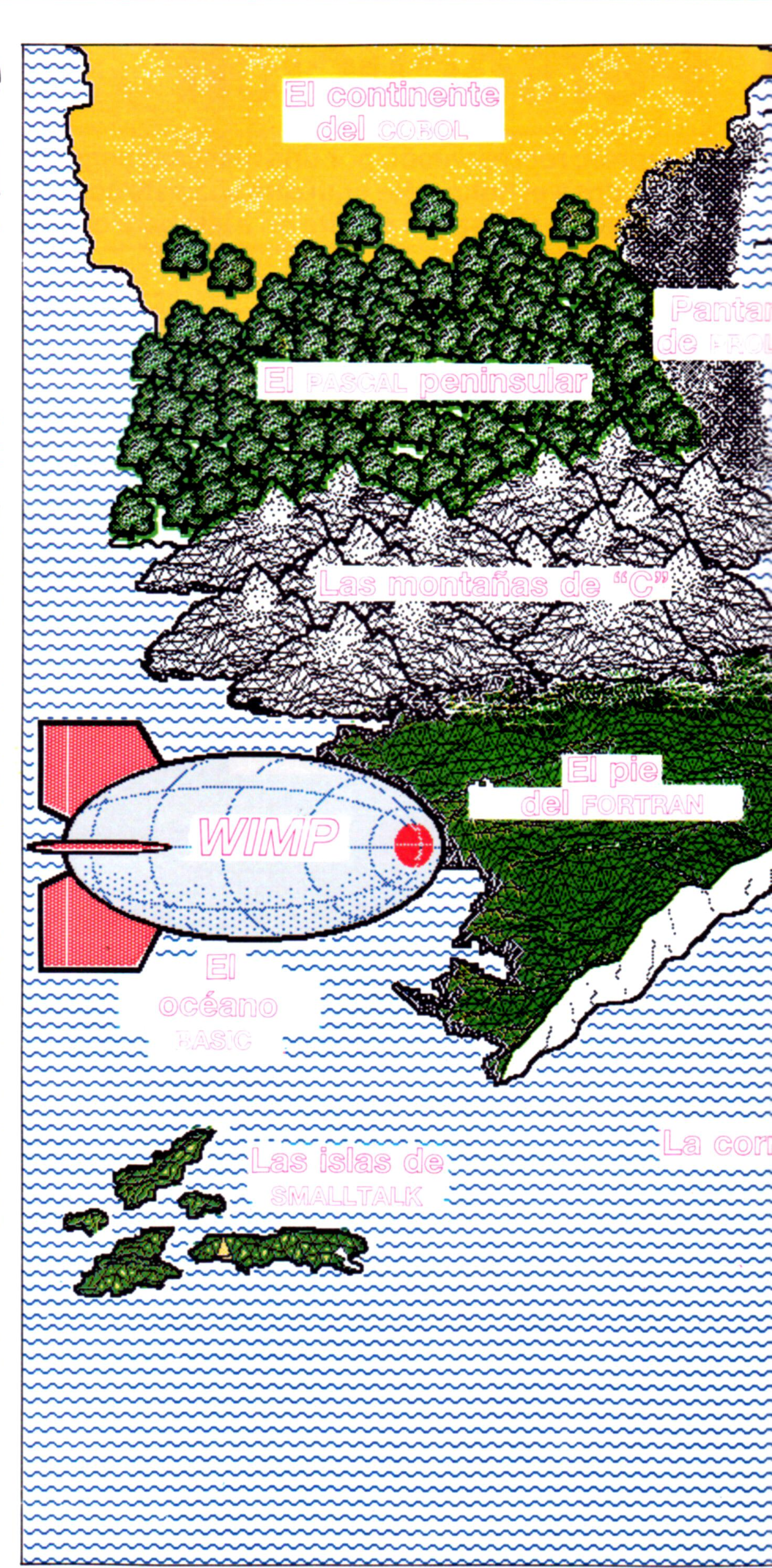
Para desplazarse utilice la palanca de mando (*joystick*) o las teclas de control del cursor.




```
10 REM ***************
20 REM * PERSECUCION *
30 REM ***************
40 CLEAR ,,2
50 GOSUB 1330
60 S=0
70 N$=CHR$(32)
80 V$=GR$(1)
90 P$=GR$(0)
100 GOSUB 830
110 ON JK GOTO 180
120 D$=INKEY$
130 DH=(D$=F1$) - (D$=F2$)
140 DV= (D$=F3$) - (D$=F4$)
150 IF DH<> THEN DX=DH:DY=0
160 IF DV<>0 THEN DY=DV:DX=0
170 GOTO 210
180 ST=STICK(0)
190 DX=(ST=7) -(ST=3)
200 DY=(ST=1) - (ST=5)
210 Z=Z-0.2
220 LOCATE 0,24
230 COLOR 0
240 PRINT "TIEMPO :";INT (Z+1);
250 IF Z<0 THEN 500
260 PX=PX+DX
270 PY=PY+DY
280 C=POINT(PX*8+4,PY*8+4)
290 IF C=0 THEN 1280
300 IF C<>-4 THEN PX=XP:PY=YP
310 LOCATE XP,YP
320 PRINT N$;
330 LOCATE PX,PY
340 COLOR 1
350 PRINT P$;
360 YP=PY
370 XP=PX
380 VX=VX+CX
390 VY=VY+CY
400 IF SCREEN(VX,VY)<>32 THEN GOSUB 670
410 IF SCREEN(VX,VY)<>32 THEN 380
420 LOCATE XV,YV
430 COLOR 0
440 PRINT N$;
450 LOCATE VX,VY
460 PRINT V$;
470 XV=VX
480 YV=VY
490 GOTO 110
500 IF INKEY$<>"" THEN 500
510 IF R<S THEN R=S
520 COLOR 0
530 LOCATE 10,6
540 PRINT "TIEMPO TRANSCURRIDO";
550 LOCATE 10,10
560 PRINT "PUNTOS :";S;
570 LOCATE 10,14
580 PRINT "PUNTUACION :";R;
590 LOCATE 10,18
600 PRINT "OTRA ?";
610 D$=INKEY$
620 IF D$="" THEN 610
630 IF D$<>"N" THEN RUN
640 SCREEN 4,6,6
650 CLS
660 END
670 DT=DT+1
680 GOSUB 780
690 IF SCREEN(XV+CX,YV+CY)=32 THEN VX=XV
    +CX:VY=YV+CY:RETURN
700 DT=DT-2
710 GOSUB 780
720 IF SCREEN(XV+CX,YV+CY)=32 THEN VX=XV
    +CX:VY=YV+CY:RETURN
730 DT=DT-1
740 GOSUB 780
750 VX=XV+CX
760 VY=YV+CY
770 RETURN
780 IF DIT>4 THEN DT=DT-4
790 IF DIT<1 THEN DT=DT+4
800 CX=(DT=1)-(DT=3)
810 CY=(DT=2)-(DT=4)
820 RETURN
830 CLS
840 COLOR 5
850 FOR VX=0 TO 39
860 LOCATE VX,0
870 PRINT CHR$(127);
880 LOCATE VX,23
890 PRINT CHR$(127);
900 NEXT VX
910 FOR VY=1 TO 22
920 LOCATE 0,VY
930 PRINT CHR$(127);
940 LOCATE 39,VY
950 PRINT CHR$(127);
960 NEXT VY
970 COLOR 4
980 FOR VX=1 TO 150
990 GOSUB 1240
1000 LOCATE PX,PY
1010 PRINT CHR$(127);
1020 NEXT VX
1030 GOSUB 1240
1040 VX=PX
1050 VY=PY
1060 COLOR 0
1070 LOCATE VX,VY
1080 PRINT V$;
1090 XV=VX
1100 YV=VY
1110 GOSUB 1240
1120 COLOR 1
1130 LOCATE PX,PY
1140 PRINT P$;
1150 XP=PX
1160 YP=PY
1170 Z=30
1180 CX=0
1190 CY=0
1200 DX=0
1210 DY=0
1220 DT=0
1230 RETURN
1240 PX=INT (RND*38)+1
1250 PY=INT (RND*22)+1
1260 IF SCREEN (PX,PY) <> 32 THEN 1240
1270 RETURN
1280 FOR I=1 TO 5
1290 PLAY "T15REDO"
1300 NEXT I
1310 S=S+1
1320 GOTO 100
1330 SCREEN 4,3,3
1340 CLS
1350 DEFINT A-Y
1360 DEFGR$ (0)=28,28,200,62,9,28,20,20
1370 DEFGR$ (1)=0,126,255,153,255,126,60,0
1380 FI$=CHR$(8)
1390 F2$=CHR$(9)
1400 F3$=CHR$(11)
1410 F4$=CHR$(10)
1420 ATTRB 1,1
1430 LOCATE 10,10,0
1440 PRINT "JOYSTICK ?";
1450 D$=INKEY$
1460 PX=RND
1470 IF D$="" THEN 1450
1480 IF D$="0" THEN JK=1
1490 ATTRB 0,0
1500 RETURN
```

La joya de un sistema

En esta nueva serie estudiaremos los sistemas WIMP, centrando nuestra atención en los "lenguajes orientados hacia el objeto"



Al recién iniciado en informática se le suele introducir en el uso de los ordenadores personales mediante programas que se ejecutan, o bien se "cargan", al poner en marcha la máquina. En el primer caso, las máquinas se proporcionan con una versión de BASIC residente, con lo que a menudo los usuarios ingenuos tienen la impresión de que los ordenadores son poco más que "máquinas de BASIC" en la que las instrucciones se entran en modalidad directa (es decir, sin números de línea de programa).

Un intérprete de BASIC residente, como el que acabamos de describir, puede ofrecer un entorno más amable que el segundo tipo (el *programa de sistema*), que no es un entorno de lenguaje empaquetado sino que se conoce como el monitor, o sistema operativo (OS). Ejemplos típicos de sistemas operativos estándares son CP/M, MS-DOS, UNIX, etcétera. Estos sistemas tienden a ser bastante hostiles y se remontan a los días en que se esperaba que los usuarios de ordenadores fueran expertos en los intrincados detalles de su máquina.

Como sabrán la mayoría de los lectores, las máquinas como el Apple Macintosh han modificado de forma radical nuestras expectativas acerca de lo que podemos esperar de ellas y especialmente nuestra interacción con las mismas. Tales sistemas tienden a basarse en el uso de ventanas, iconos y ratones, y por este motivo han llegado a conocerse como sistemas WIMP (*windows, icons and mice*). La aparición de la tecnología WIMP es apenas un aspecto de un campo de investigación que es relativamente poco conocido (al cual se suele aludir como *programación orientada hacia el objeto*) y el trabajo que se está realizando en este campo tiene enormes implicaciones de cara al futuro. Es esta investigación, y los cambios a los que ha dado lugar, lo que examinaremos a lo largo de esta serie.

El lenguaje más "objetivo" que se ha desarrollado hasta ahora es, sin lugar a dudas, el SMALLTALK, producto de un equipo de investigadores del Centro de Investigación de Xerox en Palo Alto (PARC). En consecuencia, comenzaremos nuestra serie con un análisis de los antecedentes de SMALLTALK y un proyecto asociado con el mismo: el Dynabook de Alan Kay.

Quizá resulte extraño que una empresa que obtiene la mayor parte de sus beneficios a partir del papel se mostrara interesada en el concepto de un entorno de oficina sin papel: la "oficina electrónica". Sin embargo, esto fue efectivamente así en el caso de la Xerox Corporation de Estados Unidos a



comienzos de los años setenta. Su previsión, al anticiparse a las futuras tendencias, ha venido a significar que Xerox, tal vez en mayor medida que ninguna otra empresa, fuera responsable de los orígenes de lo que ahora conocemos como sistemas WIMP.

Xerox comprendió que con la intensa competencia de IBM, DEC y otras compañías, su tardía entrada en el campo de la informática sería un serio inconveniente a menos que pudiera captar una parte del mercado de fabricantes de ordenadores bien establecidos. Dieron el arriesgado paso de crear el PARC, dando a su equipo de investigadores y técnicos un extenso informe y un gran presupuesto para investigar en el campo de la automatización de oficinas, la inteligencia artificial (AI), la interface a la medida del hombre (MMI) y los sistemas de ordenador en general.

El punto de partida para el SMALLTALK fue el futurista Dynabook, o sistema de referencia para todos. Éste era la invención de Alan Kay, entonces estudiante de investigación, quien imaginó una época en la que todo el mundo tendría un pequeño ordenador autoalimentado (del tamaño aproximado de un libro) que permitiría a cada usuario el acceso dinámico a una enciclopedia completa para consulta de conocimientos.

El Dynabook, portátil y de altas prestaciones, tendría una visualización en alta resolución, dispositivos de entrada y salida para comunicaciones tanto visuales como de audio, y conexiones de radio a una red de satélites con una base de datos compartida. Lamentablemente, los problemas técnicos de la miniaturización, el procesamiento masivo y las fuentes de memoria requeridos para implementar la idea de Kay, dieron como resultado la dispersión del equipo de investigación Dynabook. Sin embargo, muchas de las ideas Dynabook han tenido una influencia intensa y duradera en la investigación y en el desarrollo, que ahora se han abierto camino en las máquinas disponibles actualmente.

Uno de los conceptos clave que surgieron durante las primeras etapas de desarrollo fue el del *centro de trabajo* personal. Hasta entonces la informática había significado compartir los recursos de una gran instalación central o, más recientemente, varios miniordenadores. El procesamiento de los datos era muy indirecto, con el procesamiento por lotes (*batch*), siendo la norma en grandes sistemas.

La diferencia más obvia entre este entorno y la filosofía PARC fue el tiempo de respuesta. Los sistemas convencionales suponían una espera de los resultados que en ocasiones era de varias horas. En 1972, la inmediatez implícita en el concepto Dynabook era bastante revolucionaria. Además, conceptos tales como el uso de un dispositivo señalador con aspecto de ratón, de iconos y ventanas, significó un replanteamiento radical de la forma de comunicar los datos hacia y desde el procesador central y la visualización. Ello condujo a la implementación en el firmware de potentes núcleos de gráficos exclusivos, y también al extraño lenguaje que los investigadores del PARC bautizaron como SMALLTALK.

El núcleo de la investigación en SMALLTALK giraba alrededor de la modelación de un sistema de hardware y software completo basado en la comunicación entre "objetos" con ciertas propiedades definidas.

La geografía de los lenguajes de programación ha visto con el correr de los años el desarrollo de muchos entornos diferentes, cada uno de ellos con sus propias características y peligros ocultos para el viajero distraído. Pero en un lejano archipiélago abandonado, denominado SMALLTALK, que el mundo descubriera hace muy poco tiempo, se ha desarrollado una modalidad de transporte que brilla sobre este entorno hostil. La nave, llamada *Wimp*, navega a través de un cielo despejado, arrojando sobre cada rincón de la tierra y del mar su sombra, que es recibida con satisfacción

Ian McKinnell

Todos estamos familiarizados con varios objetos distintos que forman parte de cualquier sistema de ordenador: el teclado, la VDU y la impresora son ejemplos obvios. Cada uno de estos dispositivos de hardware posee propiedades fijas, y sólo comprende cómo realizar determinadas tareas. No tiene sentido pedirle al teclado que imprima un documento, por ejemplo. Cada objeto, a su modo, retiene una cierta cantidad de datos (información sobre sí mismo) y el conocimiento requerido para llevar a cabo una limitada cantidad de tareas correctamente. La pantalla VDU parece "saber" cómo visualizar los datos que se le envían, y ocuparse de sus propias tareas domésticas, tales como volver a enviar el cursor al lado izquierdo de la pantalla cuando el texto intenta continuar más allá del final de la línea.

En realidad, este objeto es una compleja red de interacción de hardware y software, en parte electrónica, en parte sistema operativo u otro software. Al considerar a cada uno de los componentes de un sistema de ordenador como un objeto, y al definir sus propiedades y los algoritmos necesarios para hacer que se conduzcan correctamente, un ingeniero de software puede crear un sistema basado en una red de objetos que se comuniquen entre sí. En vez de la tradicional técnica de lenguaje estructurado de llamar a un procedimiento por su nombre para procesar datos (pasándolos como sus parámetros), los "objetos" de programa se describen según ciertas clases o categorías, y contienen los métodos para procesar tipos de datos específicos. Éstos se suministran como argumentos de un "mensaje". Todo lo que hace el mensaje es pasar los datos y decirle al objeto receptor qué método aplicar: al que envía el mensaje no le concierne en absoluto cómo se lleva a cabo exactamente el método.

SMALLTALK es la implementación más completa del enfoque orientado hacia el objeto, pero requiere grandes recursos de memoria y potencia de procesamiento. Más recientemente, ha habido un interés considerable por el MODULA-2, en especial en Estados Unidos. Este lenguaje europeo es un derivado del PASCAL y fue diseñado por el creador de este lenguaje, el profesor Niklaus Wirth, para programar grandes sistemas en los que un equipo de programadores debía poder implementar distintos "módulos" de forma independiente (de ahí su nombre). Los datos, tipos de datos y procedimientos se pueden mantener separados en cualquier módulo a menos que se "importen" o "exporten" explícitamente para ser utilizados por otros módulos. Esto representa una analogía completa con la idea del SMALLTALK, con la excepción de que el concepto de pasar mensajes y argumentos de datos se sustituye por las llamadas a procedimientos y listas de parámetros de estilo PASCAL, más convencionales. Esto proporciona una forma simple y segura de construir grandes sistemas sin los peligrosos efectos colaterales a los que puede dar lugar el acceso ilimitado a todos los objetos del sistema.

Aunque todavía es muy reciente la aparición de máquinas como el Apple Macintosh, los consumidores ya están dando por sentadas la inmediatez y amabilidad de los sistemas operativos WIMP. SMALLTALK fue responsable del nacimiento de tales sistemas, y su implementación tuvo importantes implicaciones en el desarrollo que ha experimentado el hardware desde entonces.

La fórmula del éxito

Ahora nos corresponde examinar la regla de Bayes



Una época de diletantismo
En los siglos XVI y XVII, el ocio y el aprendizaje tendían a entremezclarse de modo tal que en nuestra época de especialización podría llegar a considerarse carente de toda seriedad. Así, no es sorprendente hallar que las apuestas y las matemáticas estuvieran unidas tan frecuentemente. El reverendo Bayes, que vemos en la ilustración, desarrolló una ecuación para el cálculo de posibilidades, y otros casi contemporáneos suyos (incluyendo a D'Alembert, Pascal y Fermat) se sintieron igualmente atraídos por los enigmas potencialmente rentables de las probabilidades

Tras haber analizado en el capítulo anterior la conversión de posibilidades a probabilidades, nos concentraremos ahora en la aplicación de cifras de probabilidades, que es donde el apostador puede valerse de la ayuda del micro.

Usted puede usar su ordenador para calcular las probabilidades que representan las posibilidades citadas, y los corredores de apuestas pueden utilizar esto para elaborar un buen registro de apuestas. Pero la mayoría de los micros se utilizan para detectar ganadores, y la única forma racional de hacerlo es evaluar el formulario. El problema es que hay demasiada información, lo que implica:

- Seleccionar qué evidencia es importante.
- Combinar evidencias desiguales.

Es aquí donde muestra su valía un teorema de estadística: la regla de Bayes. El reverendo Bayes era un clérigo del s. XVIII que produjo esta ecuación:

$$P(H|E)=P(E|H)\times P(H)/P(E)$$

que desde el punto de vista de las posibilidades se puede expresar así:

$$O(H|E)=O(H)\times LR(H|E)$$

Éstas pueden parecer complicadas, pero en realidad son bastante directas. En la primera ecuación, $P(H|E)$ es la probabilidad condicionada, P, de una hipótesis, H, dada (|) alguna evidencia, E.

El $O(H|E)$ de la segunda ecuación es el mismo que el $P(H|E)$, pero expresado como posibilidades (a favor). La segunda ecuación se puede leer como: las posibilidades de una hipótesis (H) dada una evidencia (E) son iguales a las posibilidades anteriores (antes de conocer E) de esa hipótesis, multiplicadas por la razón de probabilidad para esa evidencia. Enseguida analizaremos las razones de probabilidad, pero primero simplifiquemos un poco las cosas examinando un ejemplo paso a paso.

Pronósticos en el fútbol

Partiendo con algunos datos de muestra, supongamos que usted tiene registros de partidos de fútbol ya disputados y desea pronosticar los ganadores locales. Hay que destacar dos puntos: primero, cuando el pronóstico de la prensa deportiva pronostica un ganador visitante, es muy raro que el encuentro acabe con un triunfo del visitante (aunque podría terminar en empate). En segundo lugar, si la posición del equipo local (antes del partido) figura entre los nueve primeros de la tabla, por lo general el partido termina con un triunfo local. Con una muestra de 131 partidos, estas observaciones se pueden resumir como dos tablas de frecuencia:

Evidencia	Resultados	
	Ganados local	Ganados visit.
Prensa pronostica triunfo visitante	3	23
Prensa no lo pronostica	60	45
Totales	63	68

Evidencia	Resultados		
	Ganados local	Ganados visit.	Totales
Equipo local entre los nueve primeros	31	25	56
Equipo local en 9.ª posición o peor	32	43	75
Totales	63	68	131

Ahora podemos utilizar estas dos tablas de frecuencia para calcular las "razones de probabilidad" que mencionábamos antes. Por lo general la razón de probabilidad se define del siguiente modo:

$$LR(H|E)=P(E|H)/P(E|\text{not-}H)$$

Para tablas de eventualidades de dos por dos como las de arriba, podemos calcular las razones de probabilidad a partir de una tabla como la siguiente:

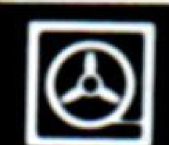
Evidencia	Resultados		
	Hipótesis	No hipótesis	Totales
	a	b	(a+b)
	c	d	(c+d)
Totales	(a+c)	(b+d)	(a+b+c+d)

$P(E|H)=$ cantidad de resultados que sustentan resultados Evidencia/Total a favor de la hipótesis; es decir, $P(E|H)=a/(a+c)$. De modo similar, $P(E|H)=b/(b+d)$. Nuestra razón de probabilidad $LR(H|E)=a/(a+c)/b/(b+d)$, con un poco de malabarismo se convierte en:

$$LR(H|E)=(a \times (b+d)/b \times (a+c))$$

Para nuestras dos tablas los resultados son éstos:

Evidencia	LR(H|E) (evidencia triunfo local)
Prensa pronostica triunfo visitante	0,1408
Prensa no pronostica triunfo visitante	1,4392
Evidencia	**LR(H|E) (evidencia gana local)**
Equipo local entre los nueve primeros	1,3384
Equipo local en 9.ª posición o peor	0,8032



Para quienes prefieren las palabras a los números, el último par de números está diciendo que si usted sabe que el equipo local está entre los nueve primeros clasificados de su tabla, las posibilidades a favor de que gane son 1,3384 veces de lo que serían si usted no supiera ese hecho. Pero si el puesto que ocupa en la tabla es el décimo u otro inferior, las posibilidades son sólo 0,8032 veces de lo que habrían sido a favor de un ganador local.

Para ver cómo se utilizan estas razones, vamos a suponer que la primera evidencia es falsa (Pronóstico <> Visitante) y que la segunda es verdadera (Poslocal <10). Comenzamos con las anteriores posibilidades a favor del local, que, dado que hay 63 locales y 68 no locales (empates o visitantes), es 63/68 = 0,9265. Esto corresponde a una probabilidad de 0,4809.

A continuación, multiplicamos las razones de probabilidad adecuadas. La primera evidencia era falsa, de modo que usamos 1,4392, y la segunda era verdadera, por lo que usamos 1,3384. Por tanto:

Posib. ulteriores $=0{,}9265\times1{,}4392\times1{,}3384$
$=1{,}7846$

Podemos volver a convertir a probabilidades con nuestra fórmula $P=F/(1+F)$, de modo que:

Prob. ulteriores $=1{,}7846/2{,}7846$
$=0{,}6409$

Esta cifra dice que hay alrededor de un 64 % de posibilidades de un triunfo local, dado que el pronóstico del diario no es un visitante y que el equipo local está situado en el puesto noveno (o uno mejor) de la tabla. Ambas evidencias eran favorables, y hemos elevado las probabilidades anteriores desde un 48 % a un 64 %. Si un corredor ofrece 6 a 4 o más, vale la pena aceptar la apuesta (pero debido al impuesto sobre las apuestas, usted aquí no puede permitirse aceptar menos que a la par).

La ventaja de la regla de Bayes es que proporciona una base racional para sopesar y combinar distintas evidencias. También es fácil de calcular: una vez que tiene las razones de probabilidad, sólo es cuestión de multiplicar. Además, dado que el resultado es en posibilidades, que se pueden convertir fácilmente a probabilidades, la regla hace muy simple ver si una apuesta es una buena inversión (a diferencia de otros sistemas que producen "números mágicos" o estimaciones de fortaleza).

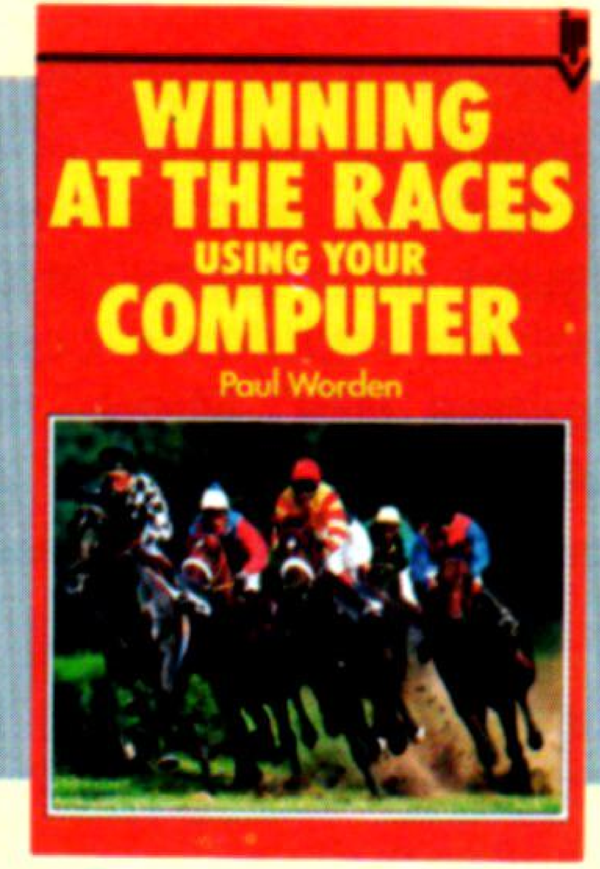
La trampa es que el método que vemos aquí probablemente producirá estimaciones exageradas si las evidencias están correlacionadas. Si el experto del diario, por ejemplo, toma en cuenta la posición en la tabla para hacer su pronóstico (como es muy probable) entonces las dos evidencias no son auténticamente independientes. Al multiplicarlas, las estamos tratando como si lo fueran.

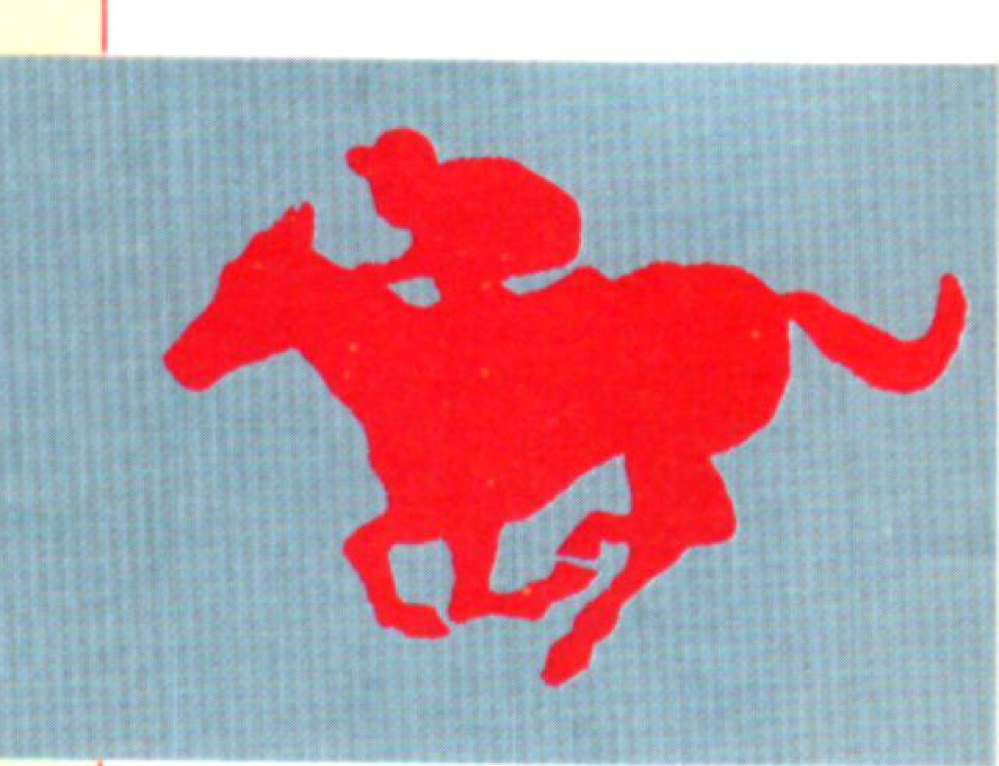
Así y todo, la regla de Bayes es muy recomendable como hilo para coser entre sí diversos indicadores de un programa de pronósticos. Si usted escribe un programa bayesiano de pronósticos, la mejor forma de minimizar el problema de la evidencia correlacionada es probar uno por uno los nuevos datos, y añadir uno nuevo sólo si mejora el rendimiento global del sistema. Si al incluir una variable plausible ésta hace que el sistema funcione menos bien, es probable que, en efecto, usted esté midiendo dos veces la misma cosa.

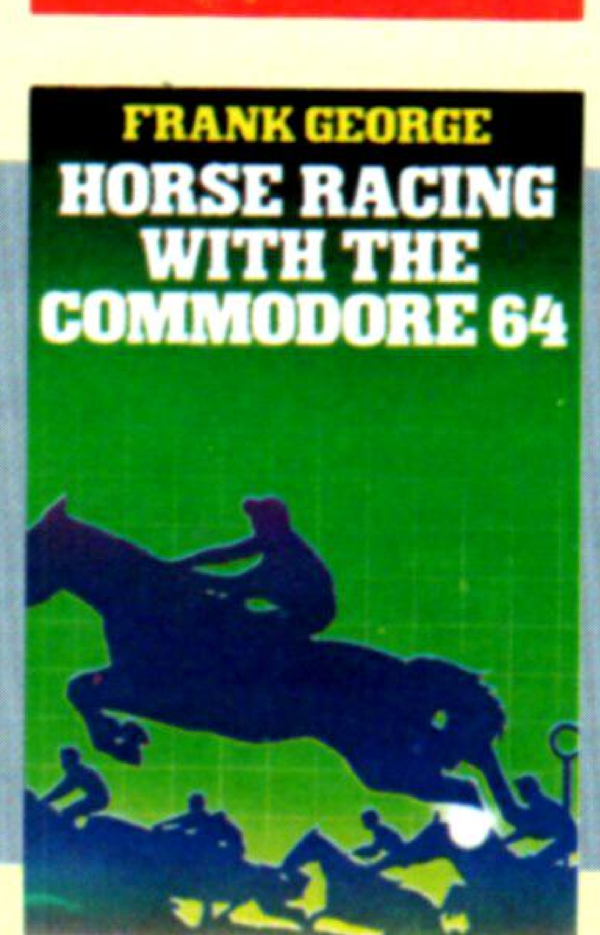
La comprobación de los métodos que emplee en una muestra de datos pasados constituye el corazón del enfoque científico a las apuestas. Lamentablemente, esto implica cierto trabajo preliminar, que la mayoría de la gente tiende a saltarse, prefiriendo, en cambio, basarse en la fe ciega. Un poco de sentido común, junto con ciertos principios estadísticos básicos, pueden ayudarlo a inclinar las posibilidades a su favor. El ordenador puede ser de ayuda en este proceso; pero una cosa que no puede darle es moderación. Ésta es esencial si ha de mantenerse fiel a sus planes, para enfocar la cuestión científicamente y no apostar hasta estar seguro y preparado.

Interesante bibliografía

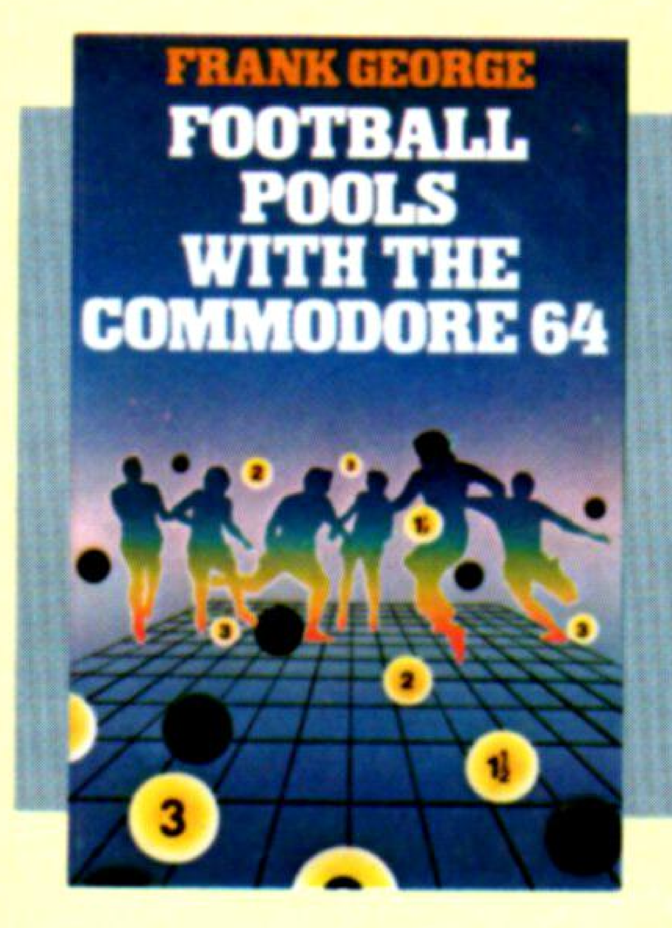
He aquí tres obras relacionadas con las apuestas y la informática. *Winning at the races using your computer* (Ganar a las carreras utilizando su ordenador), de Paul Worden, ha sido editada por Interface Publications. Las dos obras de Frank George, *Horse racing with the Commodore 64* (Carreras de caballos con el C64) y *Football pools with the Commodore 64* (Quinielas con el C64) son ediciones Collins



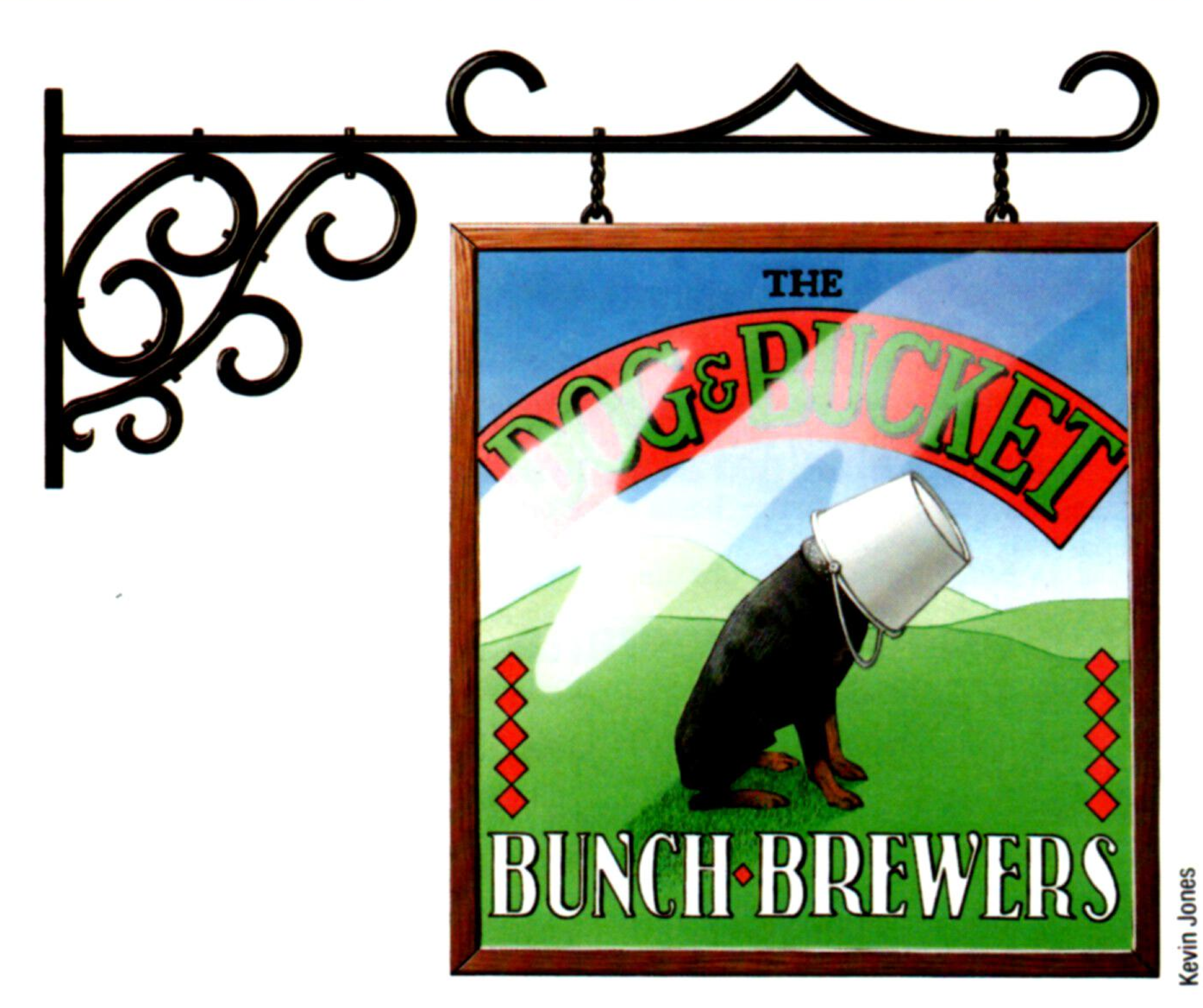












Kevin Jones

Complementos al BASIC

Algunos complementos ya se han publicado en pp. 1988 y 2085, y los lectores habrán de entrarlos cuando sea conveniente. Además, se deben insertar los siguientes complementos:

Spectrum:

```
4350 v=INT(RND*a):
     RETURN
4540 RESTORE 9920: FOR
     e=1 TO t(t,n,4): READ
     h: NEXT e: GOSUB h
4570 RESTORE 9930: FOR
     e=1 TO t(t,n,3): READ
     h: NEXTe: GOSUB h:
     RETURN
5470 RESTORE 9910: FOR
     e=1 TO (t(t,n,1)+1:
     READ h: NEXT e:
     GOTO h
9910 DATA 5480,5490,
     5500,5520,5530,5540,
     5550
9920 DATA 3930,3940,3995
9930 DATA 3030,3060,
     3070,3080,3100,3120,
     3140,3150,3160,3170,
     3180,3190,3200,3210,
     3220,3230,3240,3250,
     3270
```

BBC Micro:

```
4350 v=RND(a): RETURN
```

Commodore 64:

Suprimir de la línea 7000 en adelante e introducir las siguientes modificaciones:

```
190 DIM t(5,40,4), k(3,30),
    c(35), s(6), h(6),
    i$(23)
310 FOR n=1 TO 23: READ
    i$(n): NEXT n
4680 m$=i$(t(t,n,4)):
     RETURN
```

He aquí el listado completo de nuestro programa de personajes interactivos, que se ejecutará sin modificación alguna en la gama de ordenadores Amstrad. También incluimos complementos para las máquinas BBC Micro, Spectrum y Commodore 64

Este listado completo incorpora los restantes árboles de decisión que cubren la conciencia de objetos, la actividad general y la interacción de los personajes. El listado está impreso en dos colores: las líneas impresas en negro ya se han publicado, las líneas en verde son las nuevas adiciones.

Hay uno o dos casos de líneas que, habiendo sido publicadas anteriormente, se han modificado para dar cabida a nuestros árboles nuevos; estas líneas están impresas en color rojo.

Cuando ejecute (RUN) el programa, puede entrar el editor de caracteres en cualquier momento pulsando la tecla cero. De lo contrario, pulsando uno, dos o tres se trasladará al jugador a la sala de tertulia, al salón y a la cocina, respectivamente.

Listado del "Dog and Bucket"

Inicialización

Estas líneas preparan las variables y leen datos en las tres matrices diferentes

```
10 REM *bienvenido al Dog an Bucket*
20 REM
30 REM .........inicializar..........
40 REM
45 GOSUB 4030: REM limpia la pantalla
50 DIM 1$(3,5),b$(12,4),c$(7,11),d$(11)
60 r=1
70 PRINT "Valores por defecto (s/n)?": GOSUB
   4110: IF i$="s" OR i$="S" GOTO 90
80 GOSUB 2350:GOTO 100
90 FOR n=1 TO 7: READ c$(n,1): FOR d=2 TO 11:
   READ c$(n,d): NEXT d: NEXT n
100 FOR n=1 TO 3: READ l$(n,1): FOR e=2 TO 5: READ
    l$(n,e): NEXT e: NEXT n
110 FOR n=1 TO 12: READ b$(n,1): FOR e=2 TO 4:
    READ b$(n,d): NEXT d: NEXT n
120 FOR n=2 TO 11: READ d$(n): NEXT n
130 DEF FNb(y,z)=VAL(b$(y,z))
140 DEF FNc(y,z)=VAL(c$(y,z))
150 DEF FNm$(c$,d)=STR$((VAL(c$))-d)
160 DEF FNi$=b$(VAL(c$(c,3)),1)
180 REM prepara arboles
190 DIM t(5,40,4),k(3,30),c(35),s(6),h(6):z=0
200 REM arbol objetos
210 FOR n=1 TO 21: REM 21 nudos elección
220 READ k(1,n),t(1,n,2),t(1,n,1): NEXT n
230 REM arbol trama
240 FOR n=1 TO 22: FOR s=1 TO 4: READ t(2,n,s):
    NEXT s: READ a$: NEXT n
250 REM arbol interaccion personajes
260 FOR n=1 TO 22:FOR s=1 TO 4: READ t(3,n,s):
    NEXT s: READ a$: NEXT n
270 REM arbol actividad general
280 FOR n=1 TO 39: FOR s=1 TO 4: READ t(4,n,s):
    NEXT s: READ a$: NEXT n
290 REM arbol conciencia objeto
300 FOR n=1 TO 13: FOR s=1 TO 4: READ t(5,n,s):
    NEXT s: READ a$: NEXT n
500 REM
510 REM prueba bucle programa
520 REM
530 GOSUB 2100: GOSUB 2150: GOSUB 2240: PRINT:
    PRINT: GOSUB 1000: GOTO 530
1000 REM
1010 REM manipulador personajes
1020 REM
1030 REM comprobar si se ha pulsado tecla
1040 GOSUB 4260: IF i$<> "" THEN GOSUB 2040:
     RETURN
1050 REM procesar cada personaje uno por uno
1060 FOR c=1 TO 6: IF c$(c,9)="7" THEN
     c$(c,9)="0"
1070 IF z=c THEN GOTO 1500
1080 IF FNc(c,10)> 0 THEN
     c$(c,10)=FNm$(c$(c,10),1):GOTO 1500
```

Manipulador de personajes

Las líneas 1000 a 1500 comprueban cada personaje uno por uno y deciden si se deben o no procesar o desplazar desde un escenario a otro

```
1090 REM bandera=0 de modo que
     restablecer bandera y procesar
     personaje
1100 RESTORE: FOR n=1 TO c*10+c-1:
     READ c$(c,10): NEXT n
1110 IF FNc(c,10)=0 THEN GOTO 1500:
     REM valor defecto=0 de modo que no
     procesar
1120      REM comprueba bandera mover
1130 IF FNc(c,11)> 0 THEN c$(c,11)=
     FNm$(c$(c,11),1):GOTO 1190
1140      REM mover bandera=0 de modo
     que poner a cero bandera y trasladar
     personaje
1150 RESTORE: FOR n=1 TO c*11: READ
     c$(c,11): NEXT n
1160 IF c$(c,11)="0" THEN GOTO 1180
1170 GOSUB 2840: GOTO 1500
1180      REM recorrer los arboles
```

```
1190 GOSUB 2400: REM inicializar
     condiciones
1200 FOR t=2 TO 5: GOSUB 2430: GOSUB
     5460: NEXT t
1210 IF FNc(c,4)> 0 THEN GOSUB 5000:
     REM arbol manipulacion objetos
1500 NEXT c: GOTO 1030: REM hacer
     siguiente personaje — cuando todos
     terminados, comprobar pulsación
     tecla/hacerlo otra vez
```

Rutinas de alto nivel

Estas rutinas comprueban personajes y objetos, imprimen sus detalles en la pantalla y llevan a cabo otras funciones de fines generales

```
2000 REM
2010 REM rutina entrada
2020 REM
2030 GOSUB 4110
2040 IF (ASC(i$)< 48) OR (ASC(i$)> 51)
     THEN RETURN
2050 IF i$="0" THEN GOSUB 2320:
     RETURN
2060 r=VAL(i$): c$(7,2)=i$: RETURN
2070 REM
2080 REM impresion escenario
2090 REM
2100 PRINT l$(r,1)
2110 RETURN
2120 REM
2130 REM imprimir objetos visibles
2140 REM
2150 PRINT "Ves: ";
2160 p=0: FOR b=1 TO 12: IF
     VAL(b$(b,2))<>r GOTO 2190
2170 p=p+1: IF p>1 THEN PRINT ", ";
2180 PRINT b$(b,1);
2190 NEXT b
2200 IF p=0 THEN PRINT "nada";
2210 PRINT: RETURN
2220 REM
2230 REM imprime personajes visibles
2240 REM
2250 p=0: FOR c=1 TO 6: IF
     VAL(c$(c,2))<> r GOTO 2290
2260 p=p+1: IF p=1 THEN PRINT "Estas
     en compañia de: ";: GOTO 2280
2270 PRINT ", ";
2280 PRINT c$(c,1);
2290 NEXT c
2300 IF p=0 THEN PRINT "Aqui no hay
     nadie".
2310 RETURN
2320 REM
2330 REM subrutina inicializar personajes
2340 REM
2350 PRINT: h=c: RESTORE: FOR c=1 TO
     7: READ c$(c,1): GOSUB 4070: PRINT
     c$ (c,1);" — ";: PRINT "Establecer este
     personaje?": GOSUB 4110
2360 IF (i$ <> "s") and (I$<>"S") THEN
     FOR n=2 TO 11: READ n$: NEXT n:
     GOTO 2390
2370 FOR d=2 TO 11: READ s$: PRINT
     d$(d);: INPUT i$: IF i$<>"" THEN c$
     (c,d)=i$
2380 NEXT d
2390 NEXT c: c=h: RETURN
2400 REM
2410 REM condiciones
2420 REM
2430 h=FNc(c,8): i=FNc(c,3) :j=FNc(c,6):
     c(1)=ABS(i> o): c(2) = ABS((FNb(j,2)
     = FNc(c,2)) AND (q=1)): c(3) =
     ABS(b$(i,3) = "s"): c(4)=ABS(i=j:
     c(5) = ABS(b$(i,4)= "s")
2440 c(6) = ABS(i=3): c(7) =
     ABS(FNc(c,5)> 5): c(8) =
     ABS(FNc(c,5)> 2): c(9) =
     ABS(VAL(c$(c,9)) = 1): c(10) =
     ABS(FNc(x,3) = 0):c(11)=ABS(FNc)
     (h,2) = FNc(c,2)): c(12) = 255
2450 c(13) = ABS(z = c): c(14) =
     ABS(g=c): c(15) = ABS(z = 0): c(16)
     = ABS(s(c)=255): c(17) =
     ABS(FNc(z,2)=FNc(c,2)): c(18) =
     ABS(h(c) = 255): c(19) = ABS(i = 2)
2460 c(20) = ABS(FNc(x,4)<0):
     c(21)=ABS(z = x): c(22) =
     ABS(FNc(x,3) = FNc(c,6)): c(23) =
     ABS(FNc(x,5)>15): c(24) =
     ABS(FNc(x,9)=7): c(25) =
     ABS(FNc(c,5)>15): c(26) =
     ABS(FNc(c,5)>17)
2470 c(27) = FNc(c,9): c(28) =
     ABS(FNc(c,2) = r): c(29) =
     ABS(FNc(c,2) = 1):
     c(30)=ABS(FNc(c,5)< 5)
2500 RETURN
2510 REM
2520 REM comprueba escenario en busca
     objetos
2530 REM
2540 f=0: REM establecer a cero 'bandera
     hallado'
2550 FOR b=1 TO 12
2560 IF FNb(b,2)=FNc(c,2) THEN
     f=1:b=12
2570 NEXT b
2580 IF f=1 THEN n=3: GOTO 2600
2590 n=39
2600 RETURN
2610 REM
2620 REM comprueba presencia del
     propietario del objeto: si esta presente,
     establece x al numero del personaje:
     salta al arbol
2630 REM
2640 f=0:x=0
2650 FOR m=1 TO 6
2660 IF (FNc(m,2)=FNc(c,2)) AND
     (FNc(m,6)=FNc(c,3)) THEN f=1:
     x=m: GOSUB 2430: m=6
2670 NEXT m
2680 IF f=1 THEN n=15: GOTO 2700
2690 n=39
2700 RETURN
2710 REM
2720 REM selecciona objeto al azar en
     escenario del personaje
2730 REM
2740 b=0
2750 FOR s=1 TO 12
2760 IF FNb(s,2)<> FNc(c,2) THEN GOTO
     2780
2770 GOSUB 4180: IF q=1 THEN b=s:
     s=12
2780 NEXT s
2790 IF b=0 THEN GOTO 2750
2800 RETURN
2810 REM
2820 REM desplaza un personaje
2830 REM
2840 IF FNc(c,4)<1 THEN RETURN: REM
     demasiado debil para moverse
2850 y=0: f=0: FOR w=2 TO 5: IF
     l$(VAL(c$(c,2)),w)="0" THEN GOTO
     2910
2860 GOSUB 4180: IF q=1 THEN f=1:
     c$(c,9)="7": GOTO 2880
2870 GOTO 2910
2880 IF FNc(c,2)=r THEN PRINT c$(c,1);"
     sale de la habitacion...";: y=1
2890 c$(c,2)=1$(VAL(c$(c,2)),w): w=5: IF
     FNc(c,2)=r THEN PRINT c$(c,1);"
     entra en la habitacion...";: y=1
2900 IF y=1 THEN y=c: GOSUB 2250:
     c=y:PRINT: PRINT: REM actualiza
     mensaje personajes presentes
2910 NEXT w: IF f=0 GOTO 2850
2920 RETURN
```

Tablas de acción

Las líneas 3000 a 3999 retienen las rutinas que se llaman durante la ejecución de los distintos árboles de decisión

```
3000 REM
3010 REM tabla de accion
3020 REM
3030 FOR n=1 TO 3: GOSUB 4090: NEXT
     n:m$=c$(c,1)+" vuelve a mirar el
     cuerpo y de pronto comprende la
     espantosa verdad. "+ CHR$(34)+ "La
     empanada está elaborada con alimento
     para gatos"+CHR$(34)+
     " ": GOSUB 4630
3040 GOSUB 4390: m$=m$+ "grita, y en
     una loca carrera las personas reunidas
     se arremolinan sobre la barra y atacan a
     Fred el barman, quien les suplica en
     vano que tengan piedad de su
     miserable vida.": GOSUB 4630: GOSUB
     4720
3050 FOR n=1 TO 2000: NEXT n:GOSUB
     4720: m$="...y al dia siguiente a Fred
     el barman no se le ve por ninguna
     parte. No obstante, hay muchisimas
     empanadas de extraña forma para
     alimentar a la siempre famelica clientela
     del Dog and Bucket...": GOSUB 4630:
     GOSUB 4720: END
3060 h(c)=255: GOSUB 4680:
     m$=c$(c,1)+ m$: GOSUB 4630:
     GOSUB 4720: RETURN
3070 z=c: GOSUB 4680: n$=c$(c,1)+m$:
     GOSUB 4300: m$=n$+m$+
     "estomago, e inmediatamente muere.
     Los otros personajes estan demasiado
     concentrados en sus bebidas como
     para darse cuenta...": GOSUB 4630:
     GOSUB 4720: g=0: c$(c,4)="-1":
     RETURN
3080 c$(c,4)="10": g=0:IF t(t,n,4)>0
     THEN GOSUB 4680: m$=c$(c,1)+m$:
     GOSUB 4630: GOSUB 4720
3090 RETURN
3100 a=20: GOSUB 4350:IF v<> 5 THEN
     RETURN
3110 GOSUB 4680: GOSUB 4630: GOSUB
     4720: RETURN
3120 a=2: GOSUB 4350: IF v<> 0 THEN
     RETURN
3130 m$=c$(c,1)+ "intenta
     reanimar"+c$(x,1)+ "sin
```

```
        conseguirlo...": GOSUB 4630: GOSUB
        4720: RETURN
3140 GOSUB 4680: m$ = CHR$(34)
        +m$+CHR$(34)+" dice"+c$(c,1)+"
        a"+c$(x,1): GOSUB 4630: GOSUB
        4720: RETURN
3150 GOSUB 4680: m$=CHR$(34)
        +m$+CHR$(34)+", "+c$(c,1)+"
        pregunta" +c$(x,1): GOSUB 4630:
        GOSUB 4720: RETURN
3160 GOSUB 4680: m$=c$(c,1)+" y
        "+c$(x,1)+m$:GOSUB 4630: GOSUB
        4720: c$(c,5)="10": RETURN
3170 GOSUB 4680: m$ =
        CHR$(34)+m$+CHR$(34) +
        "dice"+c$(c,1)+ "a "+c$
        (x,1):GOSUB 4630: GOSUB 4720:
        RETURN
3180 GOSUB 4680: m$=c$(c,1)+m$:
        GOSUB 4630: GOSUB 4720:
        c$(c,5)+"5": RETURN
3190 GOSUB 4680: m$=c$(c,1)+m$:
        GOSUB 4630: GOSUB 4720: RETURN
3200 GOSUB 4680: m$ = CHR$(34)
        +m$+CHR$(34)+" dice"+c$(c,1)+"
        afligi-
        damente...": GOSUB 4630: GOSUB
        4720: c$(c,5)="10": RETURN
3210 GOSUB 4680: m$=CHR$(34)
        +m$+CHRS(34)+" dice "+c$(c,1):
        GOSUB 4630: GOSUB 4720: c$
        (c,9)="3": RETURN
3220 x=FNc(c,8): m$=c$(c,1)+" y"
        +c$(x,1)+ " estan enfrascados en una
        conversacion.":GOSUB 4630: GOSUB
        4720: x=0: RETURN
3230 x=FNc(c,3): IF x>0 THEN GOSUB
        4680: m$=c$(c,1)+ "examina
        ebriamente "+ b$(x,1)+ m$: GOSUB
        4630: GOSUB 4720: GOSUB 4220: x=0
3235 RETURN
3240 GOSUB 4680: m$ = CHR$(34)
        +m$+CHR$(34)+" se
        queja"+c$(c,1): c$(c,5)="4": GOSUB
        4630: GOSUB 4720: GOSUB 4220:
        RETURN
3250 GOSUB 4680: m$ =
        CHR$(34)+m$+CHR$(34) + "grita
        "+ c$(c,1)+", mirando la ": GOSUB
        4630: IF FNc(c,3)=3 THEN
        m$="empanada.": GOSUB 4630:
        GOSUB 4720: GOSUB 4220: RETURN
3260 m$="bocadillo.": GOSUB 4630:
        GOSUB 4720: GOSUB 4220: RETURN
3270 GOSUB 4680: m$=c$(c,1)+m$:
        GOSUB 4630: GOSUB 4630:
        m$="bebida": GOSUB 4630: GOSUB
        4720: RETURN
3900 REM
3910 REM gosub tabla
3920 REM
3930 s(c)=255: m$=c$(c,1)+" se arrodilla
        junto al cuerpo postrado de "+
        c$(z,1)+". La horrible verdad toma
        cuerpo lentamente, pero los demas
        parecen estar demasiado borrachos
        como para prestarle ninguna atencion
        inmediata...": GOSUB 4630: GOSUB
        4720: RETURN
3940 f=0: x=0: FOR y=1 TO 6
3950 IF y=c THEN GOTO 3970
3960 IF FNc(y,2)=FNc(c,2) THEN f=1:
        GOSUB 4180:IF q=1 THEN x=y:
        GOSUB 2430: y=6
3970 NEXT y: IF f=0 THEN t(3,1,4)=2:
        RETURN
3980 IF x=0 THEN GOTO 3940
3990 RETURN
3995 c$(c,5)=FNm$(c$(c,5),1):
        RETURN
```

Rutinas de bajo nivel
Sacan datos a la pantalla y se ocupan de otras funciones de bajo nivel, tales como hacer sonar un BEEP y tomar caracteres del teclado

```
4000 REM
4010 REM subrutinas del sistema de bajo
        nivel
4020 REM
4030 REM limpia la pantalla
4040 REM
4050 CLS: RETURN
4060 REM
4070 REM beep
4080 REM
4090 PRINT CHR$(7);: RETURN
4100 REM
4110 REM tomar un caracter del teclado
4120 REM
4130 i$=INKEY$: IF i$="" GOTO 4130
4140 RETURN
4150 REM
4160 REM rutina numero aleatorio
4170 REM
4180 q=INT(RND(1)*2)+1: RETURN
4190 REM
4200 REM poner a cero codigos personajes
4210 REM
4220 c$(c,8)="0": c$(c,9)="0": RETURN
4230 REM
4240 REM comprobar si tecla pulsada
4250 REM
4260 i$=INKEY$: RETURN
4320 REM
4330 REM rutina numero aleatorio variable
4340 REM
4350 v=INT(RND(2)*a): RETURN
4360 REM
4370 REM imprimir el/ella
4380 REM
4390 IF c$(c,7)="m" THEN m$="ella ":
        RETURN
4400 m$="él ": RETURN
4500 REM
4510 REM bloque de saltos
4520 REM
4530 REM nudos reentrantes
4540 ON t(t,n,4) GOSUB 3930,3940,3995
4550 RETURN
4560 REM nudos de accion
4570 ON t(t,n,3) GOSUB 3030, 3060, 3070,
        3080, 3100, 3120, 3140, 3150, 3160,
        3170, 3180, 3190, 3200, 3210, 3220,
        3230, 3240, 3250, 3270: RETURN
4600 REM
4610 REM imprimir mensajes si el jugador
        esta presente
4620 REM
4630 IF FNc(c,2)=r THEN PRINT m$;
4640 m$="": RETURN
4650 REM
4660 REM selecciona un mensaje de la
        sentencia data
4670 REM
4680 RESTORE 7030: FOR m=1 TO t(t,n,4):
        READ m$: NEXT m:RETURN
4690 REM
4700 REM imprime una linea en blanco
4710 REM
4720 IF FNc(c,2)=r THEN PRINT: PRINT
4730 RETURN
```

Recorrido del árbol
Las líneas 5000 a 5999 clasifican los distintos árboles. Aquí también se incluyen las rutinas para el árbol de manipulación de objetos

```
5000 REM rutinas arbol objetos
5010 p=0: REM poner a cero bandera
        imprimir
5020 IF FNc(c,2)=r THEN p=1
5030 n=1: REM empezar en el nudo 1
5040 IF n> 21 GOTO 5070
5050 k=c(k(1,n))+1: IF k(1,n)=12 THEN
        GOSUB 4180: k=q
5060 n=t(1,n,k): GOTO 5040
5070 IF n>=24 GOTO 5090
5080 ON (n-21) GOSUB 2540,2640: GOTO
        5040
5090 ON (n-23) GOTO
        5100,5130,5160,5180,
        5210,5240,5260,5270,5280,5300,
        5310,5330,5340,5360,5370,5430
5100 GOSUB 2740: c$(c,3)=STR$(b)
5110 IF p=1 THEN PRINT c$(c,1);"
        recoge";b$(b,1): PRINT
5120 b$(b,2)="0": c$(c,9)="4": RETURN
5130 c$(c,3)=c$(c,6)
5140 IF p=1 THEN PRINT c$(c,1);"
        recoge";FNi$: PRINT
5150 b$(VAL(c$(c,3)),2)="0":
        c$(c,9)="4": RETURN
5160 IF p=1 THEN PRINT c$(c,1);" bebe un
        sorbo de";FNi$: PRINT
5170 c$(c,4)=FNm$(c$(c,4),-1): RETURN
5180 GOSUB 4180: IF (p=1) AND (q=1)
        THEN PRINT c$(c,1);" se esta
        comiendo el bocadillo.": PRINT
5190 c$(c,4)=FNm$(c$(c,4),-2):c$
        (c,9)="6": GOSUB 4180: IF q=
        1 THEN GOSUB 4220
5200 RETURN
5210 IF p=1 THEN PRINT c$(c,1);" muerde
        un bocado de la empanada, gruñe y la
        tira al suelo.": PRINT
5220 g=c:REM establece bandera empanada
        comida
5230 c$(c,3)="0":
        c$(c,4)=FNm$(c$(c,4),10):
        b$(3,2)=c$(c,2): RETURN
5240 IF p=1 THEN PRINT c$(c,1);"
        deja";FNi$: PRINT
5250 b$(VAL(c$(c,3)),2)=c$(c,2):
        c$(c,3)="0": RETURN
5260 c$(c,5)=FNm$(c$(c,5),-1): RETURN
5270 GOSUB 5240: RETURN
5280 IF p=1 THEN PRINT c$(c,1);"
        tira";b$(VAL(c$(c,3)),1);" a ";c$ (x,1):
        PRINT
5290 c$(x,4)=FNm$(x,4),1):
        b$(VAL(c$(c,3)),2)=c$(c,2):
```

```
        c$(x,8)=STR$(c): c$(x,9)="5":
        c$(c,3)="0": RETURN
5300 GOSUB 4220: RETURN
5310 IF p=1 THEN PRINT "Creo que tengo
        tu bebida, dice ";c$(c,1);" a ";c$(x,1):
        PRINT
5320 c$(c,8)=STR$(x): c$(c,9)="2":
        RETURN
5330 c$(c,4)=FNm$(c$(c,4),2): RETURN
5340 IF p=1 THEN PRINT c$(c,1);"le da
        ";FNi$;" a ";c$(x,1): PRINT
5350 c$(x,3)=c$(c,3): c$(c,3)="0":
        c$(c,8)=STR$(c): c$(c,9)="1":
        RETURN
5360 GOSUB 4220: RETURN
5370 IF p=0 GOTO 5420
5380 IF p=1 THEN PRINT c$(c,1);" con voz
        ebria le agradece ";c$(VAL(c$
        (c,8)),1);" que le haya devuelto su
        bebida ";
5430 RETURN
5440 REM
5450 REM clasifica arboles
5460 n=1
5470 ON (t(t,n,1)+1) GOTO
        5480,5490,5500,
        5520,5530,5540,5550
5480 k=c(t(t,n,2))+1: n=t(t,n,2+ k):
        GOTO 5470
5490 GOSUB 4530: n=t(t,n,3): GOTO 5470
5500 GOSUB 4570
5510 RETURN
5520 GOSUB 4680: GOSUB 4630: GOSUB
        4720
5530 RETURN
5540 a=t(t,n,4): GOSUB 4350:
        n=t(t,n,3)+v: GOTO 5470
5550 k=c(t(t,n,2)): n=t(t,n,3)+k: GOTO
        5470
```

Almacén de datos

De la línea 6000 en adelante se retienen los datos para las diversas matrices, así como los mensajes utilizados por los módulos del programa

```
6000 REM
6010 REM datos personajes
6020 REM
6030 DATA "Luis Cubas","2", "7", "10",
        "10", "7", "v", "0", "0", "1", "4",
        "Lola Fiestas", "1", "8", "30", "10",
        "8", "m", "0", "0", "1", "5", "Pepe
        Viñas", "1", "9", "8", "10", "9",
        "v", "0", "0", "1", "6", "Mari
        Tapas", "2", "0", "20", "10", "10",
        "m", "0", "0", "1", "5"
6040 DATA "Javi Salado","2", "11", "10",
        "6", "11", "v", "0", "0", "1", "6",
        "Gina Fizz", "1", "12", "15", "6",
        "12", "m", "0", "0", "1", "5","tu",
        "1", "0", "255", "255", "0", "v",
        "0", "0", "0", "0"
6050 REM
6060 REM datos escenarios
6070 REM
6080 DATA "estas en la sala de tertulia del
        Dog and Bucket. En un rincón hay un
        grupo de dudosos personajes jugando
        al domino. Detras del mostrador esta
        Fred el barman, con su habitual talante
        festivo. Hay salidas al este.", "0", "0",
        "2", "0"
6090 DATA "He aqui el salon del Dog and
        Bucket, al cual le vendría muy bien una
        redecoración completa. El suelo parece
        haber sido regado regularmente con
        cerveza. Hay puertas al oeste y al sur.",
        "0", "3", "0", "1"
6100 DATA "¡Aj! Esta es la cocina, donde se
        preparan las famosas empanadas de
        carne del Dog and Bucket, para una
        clientela siempre famelica. Puedes
        observar algunas latas vacias de
        alimento para gatos, lo cual no deja de
        ser extraño, ya que no hay ningun
        gato.","2","0","0","0"
6110 REM
6120 REM datos objetos (para b$(12,4))
6130 REM
6140 DATA "un vaso de cerveza", "2", "n",
        "s", " una lata vacia de alimento para
        gatos", "3", "n", "n", " una
        empanada de carne Dog and Bucket",
        "1", "s", "n", "una banqueta de bar",
        "2", "n", "n", "un cenicero", "1",
        "n", "n"
6150 DATA " un bocadillo de jamon rancio",
        "2", "s", "n", " una pinta de cerveza
        amarga", "0", "n", "s", " una crema
        de menta","0", "n", "s", "un whisky
        con soda","0", "n", "s", " un vodka
        puro", "2", "n", "s", " una pinta de
        cerveza añeja", "0", "n", "s", " una
        ginebra con ginger ale", "0", "n", "s"
6160 REM
6170 REM datos atributos personajes (para
        d$(11))
6180 REM
6190 DATA "Escenario", "Fortaleza",
        "Inventario", "Humor", "Objeto
        propio", "Sexo", "Ultimo pers (lch)",
        "Codigo ultima instruccion (lcd)",
        "Frecuencia de manipulación",
        "Frecuencia de movimiento"
6200 REM
6210 REM datos arbol objetos
6220 REM
6230 DATA 1,2,22,12,5,4,2,7,6,3,9,8,4,11,
        10,12,39,24,12,6,25,5,12,39,
        6,13,27,12, 23,29,12,30,14, 12,26,
        39,12,28,39,12,31, 17,7,18,
        16,8,39,19,9,21,
        39,10,36,20,12,33,32,12,
        35,34,11,38,37
6240 REM
6250 REM datos arbol trama
6260 REM
6270 DATA 0,13,2,22," ",0,14,5,3,"
        ",0,15,21,4," ",5,0,18,3,"
        ",0,16,6,7," ",0,17,7,11,"
        ",0,18,9,8," ",0,17,12,13,",
        "0,19,10,17," ",5,0,14,3,"
        ",1,0,7,1," ",4,0,0,0," ",2,0,1,0,"
        ",2,0,5,1," ",4,0,0,0," ",4,0,0,0,"
        ",2,0,2,4," ",2,0,3,2," ",2,0,4,3
6280 DATA " ",2,0,4,0," " 4,0,0,0,"
        ",4,0,0,0," "
6300 REM arbol interaccion
6320 DATA 1,0,3,2," ",4,0,0,0,"
        ",0,20,6,4," ",0,21,7,5," ",4,0,0,0,"
        ",5,0,8,3," ",2,0,6,0," "0,22,11,14,"
        ",0,23,12,17," ",0,24,13,18,"
        ",4,0,0,0," ",4,0,0,0," ",4,0,0,0,"
        ",5,0,15,2," ",4,0,0,0," ",2,0,7,5,"
        ",0,25,22,21," "
6330 DATA 5,0,19,2," ",4,0,0,0,"
        ",2,0,8,6," ",2,0,9,7," ",2,0,10,8,"
        "
6340 REM arbol actividad general
6350 DATA 5,0,2,5," ",0,26,11,7,"
        ",6,27,24,7," ",0,28,8,12,"
        ",0,29,9,15," ",5,0,21,3,"
        ",2,0,11,9," ",4,0,0,0," ",4,0,0,0,"
        ",4,0,0,0," ",0,30,10,18,"
        ",5,0,13,2" ",2,0,12,10," ",4,0,0,0,"
        ",5,0,16,2," ",3,0,0,11,"
        ",3,0,0,12,""
6360 DATA 5,0,19,2," ",4,0,0,0,"
        ",2,0,13,13," ",2,0,14,14,"
        ",2,0,14,15," ",2,0,14,16,"
        ",4,0,0,0," ",0,11,31,32,"
        ",4,0,0,0," ",5,0,33,3,"
        ",0,11,36,37," ",5,0,38,2,"
        ",2,0,14,17," "
6370 DATA 4,0,0,0," ",2,0,15,0,"
        ",4,0,0,0," ",2,0,16,18," ",4,0,0,0,"
        ",4,0,0,0," ",2,0,17,19," ",4,0,0,0,"
        ",2,0,18,20," "
6380 REM arbol conciencia objeto
6390 DATA 0,4,8,2," ",5,0,3,5,"
        ",4,0,0,0," ",4,0,0,0," ",4,0,0,0,"
        ",4,0,0,0," ",2,0,14,21," ",1,0,9,3,"
        ",5,0,10,4," ",2,0,19,22,"
        ",2,0,14,23," ",4,0,0,0," ",4,0,0,0,"
        "

7000 REM
7010 REM datos mensajes
7020 REM
7030 DATA "Un extraño color invade el
        aire...¿podria ser el aroma de Catty-Kit
        A La Carte? "," de pronto se desploma
        sobre el suelo apretandose el estomago
        ", " parece muy enfermo, y advierte a
        los demas que no toquen la
        empanada."
7040 DATA " examina atentamente la lata, y
        su semblante adquiere un aire
        pensativo. "," ¿Que estas haciendo
        con mi bebida? "," ¿Donde has
        estado?"
7050 DATA ", en un arrebato propio de
        borrachos, saltan sobre la mesa y se
        ponen a bailar."
7060 DATA "Pareces alegre","trepa a gatas
        sobre la barra, intenta bailar y vuelve a
        caer."
7070 DATA ", con el estupor de la
        borrachera, de pronto levanta la vista,
        se asoma por la pantalla de la VDU y te
        mira a ti...","Fred el barman sirve otra
        pinta...","Fred
        esta ocupado lavando vasos"
7080 DATA "Es una vida de perros..."
        ,"No he venido a enterrar a
        Cesar...","Deja que te cuente la
        historia de mi vida...","Barman,
        lleneme la copa!"
7090 DATA "Hola a todos..."," como si
        ocultara el secreto de la vida.", "Para
        que has hecho eso!!","Creo que me
        voy a poner enfermo!"
7100 DATA "Ahh...Este trago esta
        sublime..."," busca
        desesperadamente","¿Quien tiene mi
        bebida?"
```

Cambio de tecla

Estudiaremos las distintas maneras de realizar cambios en el teclado del Amstrad desde su sistema operativo

El sistema operativo del Amstrad tiene una sección, denominada *gestor de teclas*, que controla todos los accesos al teclado y a la barra de mando. La auténtica fuerza del gestor de teclas reside en el hecho de que el teclado es puro *soft*. Esto significa que el código obtenido de cada tecla física es definible por el usuario. Esto facilita la implantación de programas tales como el CP/M, dado que podemos adaptar las funciones de las teclas según convenga para que den los códigos deseados en lugar de modificar el programa para que acepte los códigos producidos por omisión.

Hay dos tipos de código obtenible al pulsar una tecla: un carácter único ASCII o uno indicativo (*token*) de ampliación. El indicativo de ampliación puede ser considerado como un flag que sirve para obtener una cadena de caracteres, y no un simple carácter, a través de la simple pulsación de una tecla.

El teclado es revisado cada cincuentavo de segundo en el reloj para uso general. Un mapa de estado de la tecla sirve para registrar qué teclas han sido pulsadas en la última revisión. Este mapa se emplea para asegurarse de que las teclas no sean leídas más de una vez y para determinar si se ha de repetir un carácter o no. También se dispone de un buffer que almacena los códigos que se obtienen de las teclas en el momento en que fueron detectadas; tales códigos pueden ser interpretados bien como caracteres ASCII, bien como indicativos de ampliación.

El gestor de teclas

Se puede acceder al gestor de teclas en varios niveles distintos. El nivel más bajo permite al usuario comprobar si se está pulsando una tecla física en ese momento.

El segundo nivel presenta el carácter único asignado a la tecla. Este carácter puede ser bien un valor ASCII, bien un indicativo de ampliación. Si se trata de un indicativo de ampliación, entonces es el usuario el que debe determinar la interpretación que se le ha de dar. Este nivel tiene en cuenta el estado actual de las teclas Shift y Control a la hora de determinar qué carácter ha de obtenerse.

El nivel más alto da simplemente el carácter siguiente que se ha pulsado en el teclado. Este carácter puede corresponder tanto a una pulsación de tecla única como puede ser el carácter siguiente de un indicativo de ampliación.

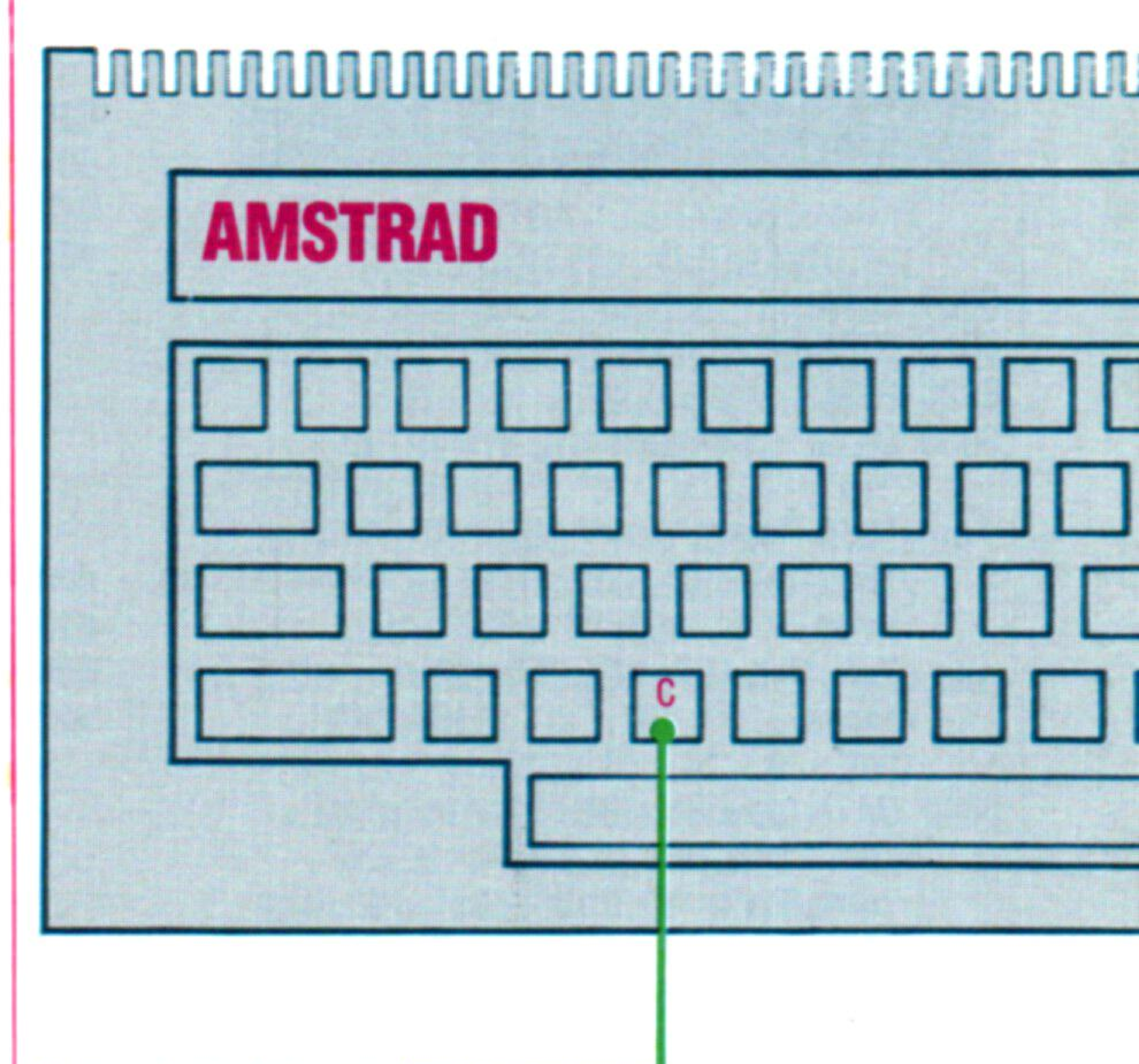
Asociadas a cada tecla existen tres tablas de traducción que determinan el código obtenido cuando se detecta una tecla pulsada en sus estados normal, *shift* o *control*. Cada una de estas entradas puede ser determinada y establecida individualmente, empleando las entradas detalladas en el diagrama. Obsérvese que los caracteres que van desde &E0 a &FC tienen un significado especial bajo BASIC, por lo que si un programa asigna una tecla para obtener uno de estos indicativos, se tendrá que reasignarlos antes de volver al BASIC.

El listado primero muestra cómo se puede establecer un conjunto de traducciones al entrar en la rutina, al mismo tiempo que se guardan sus estados iniciales.

Indicativos de ampliación

Los códigos del &80 al &9F son los denominados *indicativos de ampliación (expansion tokens)*. Cuando son encontrados se almacenan en el buffer a menos que no sean extraídos por medio de KM_READ_CHAR o KM_WAIT_CHAR. En este caso el gestor de teclas trata de determinar qué cadena se asigna a ese indicativo de ampliación e inserta los caracteres en secuencia dentro del buffer hasta el final de la cadena.

Gestión del teclado



Código de identificación
A cada tecla se le asigna un número de tecla único imposible de alterar por el firmware. El código ASCII que se obtiene mediante una tecla se determina a través de tres tablas, que proporcionan el código de la tecla en su estado normal, de *shift* o de *control*

Caroline Clayton



Las cadenas de ampliación se almacenan en un buffer, que puede ser de cualquier tamaño según el número de cadenas requerido. Cada cadena puede componerse de un máximo de 255 caracteres siempre que haya espacio suficiente en el buffer, el cual debe hallarse en los 32 K centrales de la RAM. El tamaño del buffer y su posición se establecen por medio de KM_EXP_BUFFER.

Cada indicativo de ampliación puede tener una cadena asignada mediante KM_SET_EXPAND. Los caracteres dentro de una cadena no son comprobados por el firmware, de modo que es posible obtener caracteres tales como &FC (el indicativo para Break) para una rutina. Las cadenas asignadas a un indicativo de ampliación sólo pueden ser examinadas, carácter tras carácter, mediante la entrada KM_GET_EXPAND.

Cada tecla en el teclado tiene un número asociado a ella. Este número es único y no puede alterarse por medio del firmware, obteniéndose así un modo absoluto de referencia a la tecla. Los números de tecla se detallan en el Apéndice III del manual de firmware y también se proporcionan con mucho esmero en los cartuchos de las unidades de disco del CPC 664 y del 6128.

La entrada KM_TEST_KEY es la entrada de nivel más bajo dentro del gestor de teclas. Permite al usuario averiguar si una determinada tecla ha sido mantenida en el último rastreo del teclado, a excepción de las teclas Control y Shift. Esta entrada es especialmente útil cuando se comprueba una sola tecla; por ejemplo, cuando el teclado ha sido sondeado periódicamente para que la tecla Escape dé a entender que ha de concluirse el programa.

Direcciones útiles

&BB27	KM-SET-TRANSLATE	est. una entr. normal en la tabla de trad. de teclas
&BB2A	KM-GET-TRANSLATE	lee la entr. normal actual en la tabla de trad. de teclas
&BB2D	KM-SET-SHIFT	establece una entr. en tabla de trad. para una tecla con *shift*
&BB30	KM-GET-SHIFT	lee la entr. en la tabla de trad. para una tecla con *shift*
&BB33	KM-SET-CONTROL	est. una entr. en la tabla de trad. para la tecla de control pulsada
&BB36	KM-GET-CONTROL	lee la entr. en la tabla de trad. para la tecla de control pulsada

También se proporciona una entrada exclusivamente para las palancas de mando: KM_GET_JOYSTICK. Esta entrada lee los estados de la palanca de mando en el mapa de estado de las teclas y es el método más rápido de determinar su estado actual. Por ello es especialmente apropiado para los programadores de juegos, dado que las palancas de mando deben ser leídas en todo momento con independencia de que se hayan alterado o no sus posiciones.

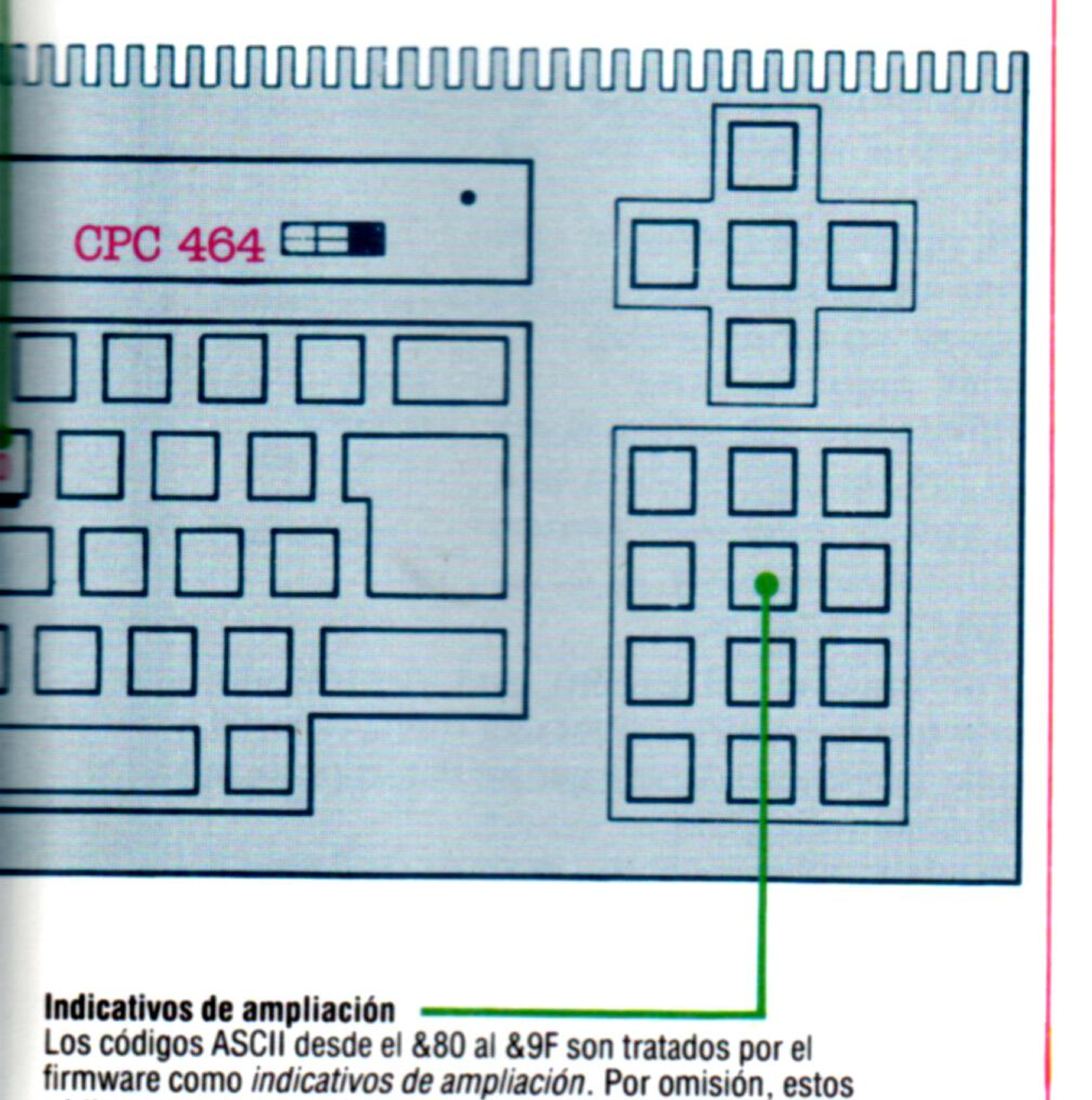
Hay dos entradas del gestor de teclas en el segundo nivel, KM_READ_KEY y KM_WAIT_KEY. La entrada KM_WAIT_KEY se emplea para dar la siguiente pulsación de tecla. Si existe una entrada en el buffer del teclado, entonces se traduce bien en un carácter, bien en un indicativo de ampliación y se retorna inmediatamente; en otro caso, la rutina espera la siguiente pulsación de tecla y da su valor traducido. KM_READ_KEY también comprueba el carácter en el buffer de teclado, y si sólo hay uno, éste se traduce; de otro modo, la rutina retorna inmediatamente, indicando mediante un flag que el buffer está vacío. KM_WAIT_KEY encuentra un uso apropiado cuando un programa no puede continuar hasta que haya recibido un input del usuario (p. ej., cuando se requiere una opción de menú). KM_READ_KEY deberá usarse cuando se sondea el teclado respecto a la pulsación de una tecla.

El tercer nivel tiene dos entradas correspondientes: KM_READ_CHAR y KM_WAIT_CHAR. La única diferencia entre éstas y las del segundo nivel está en que sólo pueden obtenerse caracteres. Si se encuentra un indicativo de ampliación, la cadena de ampliación correspondiente es extraída carácter a carácter antes de ser examinada la entrada siguiente en el buffer del teclado.

Prioridad de carácter

El gestor de teclas tiene, además del buffer de teclado, un buffer de un solo byte conocido por *prio-*

Muy sensible
El firmware mantiene un mapa de estado de las teclas que registra las teclas pulsadas en el momento de la interrupción del reloj para uso general (cada cincuentavo de segundo). Los valores de la tecla se almacenan en un buffer y el firmware permite bien comprobar la pulsación de una tecla o bien esperar esa pulsación



Indicativos de ampliación
Los códigos ASCII desde el &80 al &9F son tratados por el firmware como *indicativos de ampliación*. Por omisión, estos códigos se asignan a las teclas en el cuaderno numérico de teclas. A cualquiera de estos indicativos se les pueden asignar cadenas de ampliación. Éstas pueden tener hasta 255 caracteres de longitud y contener códigos de control

ridad de carácter (*putback character*). Este buffer es consultado antes que el buffer principal y si contiene un carácter, se le da prioridad. Esto permite colocar un carácter en cabeza dentro del buffer que puede ser utilizado, por ejemplo, para señalar como flag que ha ocurrido un determinado evento. El gestor de teclas sobreescribe este carácter prioritario cuando se detecta un Break, ya que el indicativo de Break se inserta en calidad de carácter prioritario.

El diagrama muestra los distintos niveles en los que puede obtenerse un carácter desde el gestor de teclas.

Cada tecla del teclado tiene asociados tres parámetros de repetición. El primero determina si se puede o no repetir esa tecla; el segundo especifica la demora entre el momento de detectar la pulsación de la tecla y el de permitir su repetición; el tercero es el intervalo de cada repetición. Las demoras son especificadas en múltiplos de rastreos de teclado y por ello corresponden a un intervalo de un cincuentavo de segundo.

Las determinaciones actuales de demoras pueden ser realizadas por KM_GET_DELAY y pueden alterarse mediante KM_SET_DELAY. Las demoras son válidas para todo el teclado, por lo que no es posible definir valores específicos para las distintas teclas.

El parámetro que determina si una tecla pueda repetirse o no es leído y variado mediante KM_GET_REPEAT y KM_SET_REPEAT.

Interrupciones

La tecla Escape es tratada por el gestor de teclas de modo distinto a las demás teclas. Cuando es pulsada, se llama la dirección de KM_TEST_BREAK. Esta rutina comprueba si también se pulsaron las teclas Control y Chift, en cuyo caso se realiza una restauración; de otro modo, la rutina retorna. Parcheando esta dirección con la instrucción RET es posible, entonces, evitar que dentro de un programa se produzca una restauración.

Hay también un evento que se proporciona para tratar las interrupciones. Este evento puede ser inicializado activándolo o desactivándolo a través del firmware.

Códigos especiales

&80–&9F	Indicativos de ampliacion. &80F corresponde a la cadena de ampliacion 0, y &9F a la cadena de ampliacion 32
&E0–&FC	Codigos control cursor BASIC
&FD	Conmutador mayusculas *on/off* Cambia el estado actual de la tecla mayusculas
&FF	Conmutador de Shift *on/off* Cambia el estado actual de la tecla Shift
&FF	Indicativo de ignorar Este caracter sera ignorado si se retorna desde el gestor de teclas

Traducción de teclas

Este listado proporciona dos rutinas para permitir que un programa pueda configurar el teclado de modo que provea un conjunto predefinido de códigos durante un programa y devolverlo a su estado inicial a la salida. La primera rutina, Setkeys, se llamará al entrar en el programa. Ésta rastrea una tabla que contiene una lista de números de tecla y la traducción que debe tener. El estado actual de las teclas se lee y almacena en la tabla antes de asignar un nuevo valor a una tecla. La segunda rutina lee las traducciones originales en la tabla volviendo a asignarlas a las teclas.

La tabla se establece aquí para configurar las teclas Cursor y Escape para ser utilizadas en un proceso de textos, pero puede ser ajustada a aplicaciones individuales

```
        ;SETKEYS
        get__tr:  equ  #BB2A
        set__tr:  equ  #BB27
        tend:     equ  #FE
21B329  setkey:   ld   hl,tnormal      ;establece tabla trad norm
7E      setlp:    ld   a,(hl)          ;toma numero tecla
FEFE              cp   tend            ;fin de tabla?
C8                ret  z
23                inc  hl
46                ld   b,(hl)          ;toma nueva trad
23                inc  hl
EB                ex   de,hl           ;guarda hl
F5                push af              ;guarda numero tecla
CD2ABB            call get__trans      ;lee trad actual
EB                ex   de,hl
77                ld   (hl),a          ;guarda trad antigua
23                inc  hl
EB                ex   de,hl
F1                pop  af
CD27BB            call set__trans      ;guarda nueva tecla
EB                ex   de,hl
18E9              jr   setlp           ;continua
        ;
        ;GETKEYS
        ;
21B329  getkey:   ld   hl,tnormal      ;restaura tabla norm
7E      getlp:    ld   a,(hl)          ;toma numero tecla
FEFE              cp   tend
C8                ret  z
23                inc  hl
23                inc  hl              ;apunta a area guardado
46                ld   b,(hl)          ;toma trad antigua
EB                ex   de,hl
CD27BB            call set__trans      ;restaura la antigua
EB                ex   de,hl
23                inc  hl
18F1              jr   getlp
        ;
        ;tablas traduccion teclas –almacenamiento – teclas – traduccion
        ;
421B00  tnorma:   defb 66,27,0         ;ESCAPE
000B00            defb 0,11,0          ;CURSOR UP
020A00            defb 2,10,0          ;CURSOR DOWN
080800            defb 8,8,0           ;CURSOR BACK
010900            defb 1,9,0           ;CURSOR FORWARD
FE                defb tend            ;end
```

Por omisión, el evento está desactivado, de modo que se habrá de crear una rutina especial que puede llamarse cada vez que se detecta una pulsación de la tecla Break.

Si se devuelve el control después de llamar KM_TEST_BREAK, el gestor del teclado comprueba si el evento de *break* está desactivado. Si lo está, en el buffer se inserta el carácter &EF, y se dispara el evento de *break*. El puntero permite que el evento se nivele con el buffer del teclado hasta el punto donde se encontró el *break*.

Con este análisis del gestor de teclas damos por concluida nuestra serie sobre el sistema operativo del Amstrad CPC.



Con este fascículo se han puesto a la venta las tapas correspondientes al noveno volumen



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 10, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:

1 **Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.**

2 **Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.**

3 **Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.**

Cada sobre de transferibles contiene una serie completa de números, del 1 al 10, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

PETICION DE FASCICULOS ATRASADOS

Si desea recibir algún fascículo atrasado o tapas, según las condiciones establecidas en el recuadro de la segunda página de cubierta («servicio de suscripciones y atrasados»), basta que rellene en LETRAS MAYUSCULAS este boletín y lo envíe a Editorial Delta, S.A., Aribau, 185, 1.°, 08021 Barcelona.

NOMBRE
APELLIDOS
FECHA NACIMIENTO, DIA ☐☐ MES ☐☐ AÑO ☐☐☐☐
PROFESION
DOMICILIO
N.° ☐☐☐ PISO ☐☐ ESCALERA ☐☐
CODIGO POSTAL
POBLACION
PROVINCIA

OBRA:
N.os de fascículos atrasados que desea recibir:
....
....
N.° de tapas: ....
....



Ya están a su disposición, en todos los quioscos y librerías, las tapas intercambiables para encuadernar 12 fascículos de

mi COMPUTER

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda



